ECSE 323 Digital System Design

Lab #4 – VHDL for Sequential Circuit Design

Group 27

Lulan Shen (260449509)
Loren Lugosch (260404057)

Part 1: Square Wave Generator Circuit

Circuit Description and Design

The square wave circuit takes as input a 50MHz clock, a reset signal, a note number between 0 and 15, an octave between 0 and 7, and a volume between 0 and 4194303.

Our music box plays square waves by calculating the appropriate period for the given note (0 = C, 1 = C#, etc.) in 50 MHz clock cycles (exponentiator circuit) and setting a signal high for two cycles during that period. That signal is the input to a T-flipflop, which toggles output when input is high; the output is high for half of the period and low for the other half, generating the square wave control signal. To make the square wave itself, the circuit sets the output to +volume when the control signal is high and –volume when the signal is low.

Humans hear a flat tone when sound with a simple sine waveform hits our ears. Square wave sounds have more harmonics in addition to the fundamental frequency of the waveform, so they sound more “interesting” to us. Square waves are also easier to make than sine waves (which require, in addition to the steps above, looking up the appropriate value of the sine function): hence their use in electronic music.

When one note’s frequency is twice that of another, we hear the same note, but an octave higher. When the octave input to the circuit is 3, for instance, we need to divide the period by 2^3 (barrel-shifter).

When reset is high, the internal signal controlling the counter in the circuit is set to zero, which in turn sets the count to zero, which in turn holds square at + or -volume. After the filtering in the other components of the testbed removes the DC component from square, the audio resulting from the circuit during reset high is silence.

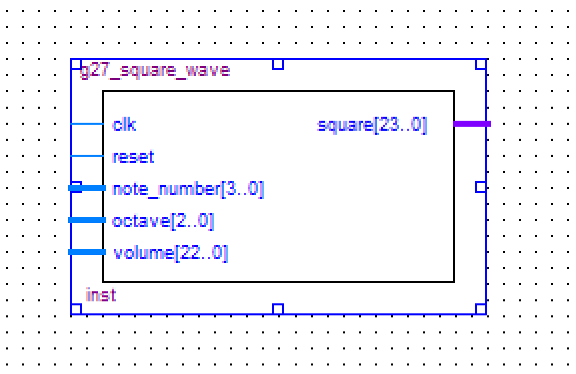


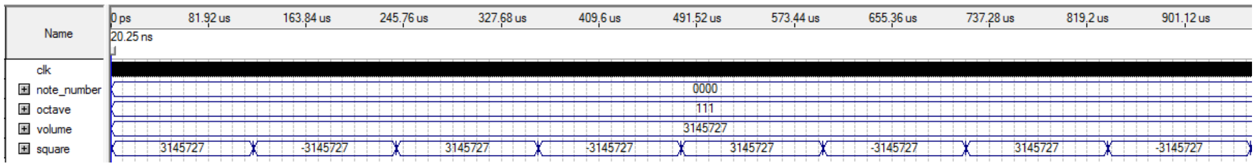
Figure 1- Symbol for Square Wave Generator

Inside the circuit, we need a counter so that we know during how many clock cycles per period the square wave controller should be toggled. Rather than doing this the tedious way required in lab 3 (wiring up an lpm counter with a NOR gate and an std_logic_vector corresponding to the division factor), we let Quartus synthesize a counter for us using the VHDL process block. The process block counts down from a number by subtracting 1, so we need an integer, “count”, and we need to specify its range. Since the division factor is 20 bits, the maximum value of count = 2^20 – 1 = 1048575, so count should be an integer of range 0 to at least 1048575.

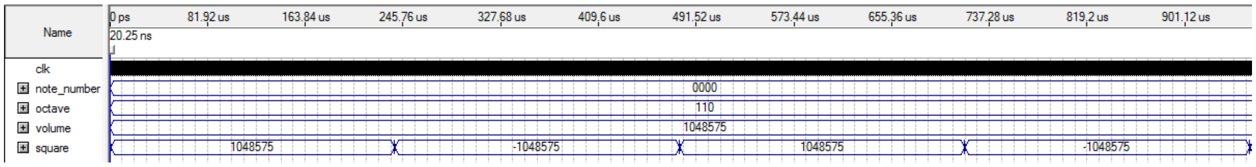
Testing and Simulation

We tested the square wave generator using several different notes, octaves, and volumes.

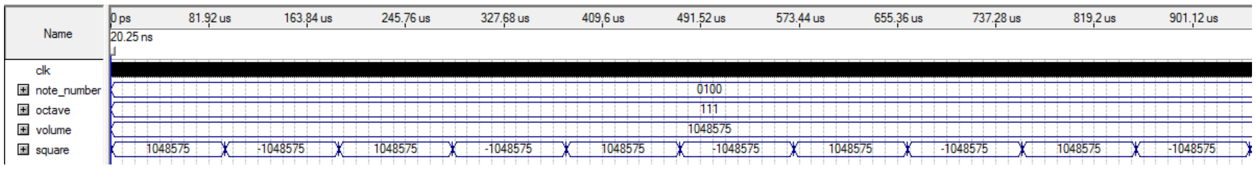
- Change volume from “000 1111 1111 1111 1111 1111” (1048575) to “010 1111 1111 1111 1111 1111” (3145727):



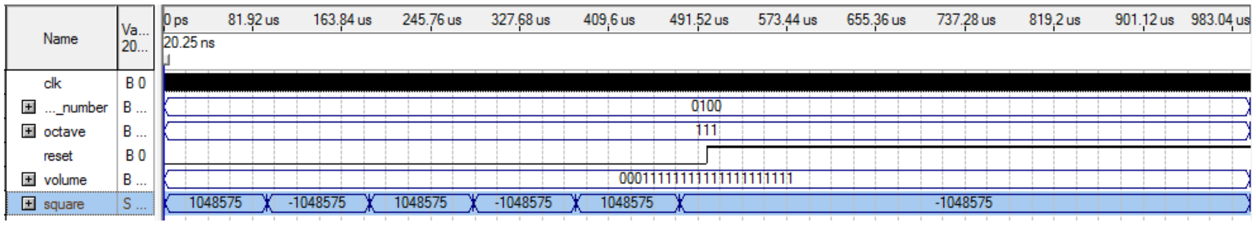
- Change octave from 111 to 110:



- Change note_number from 0000 to 0100:



- Assert reset:



When we were satisfied with the simulation results, we put together the testbed using the provided decimator and audio interface.

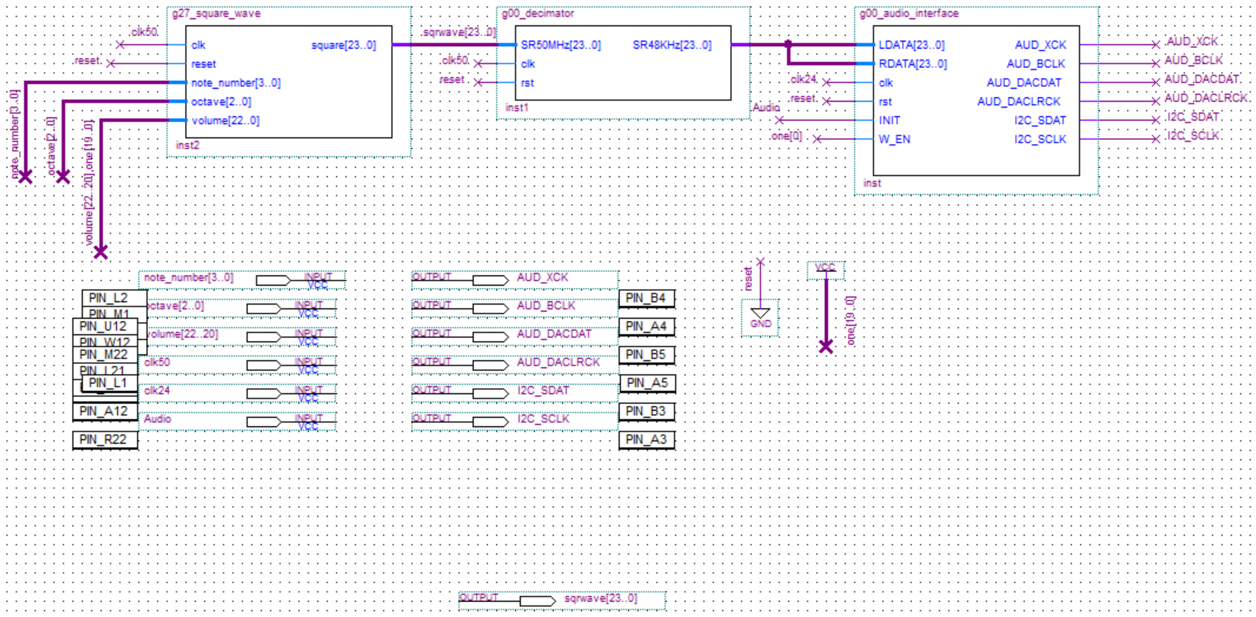


Figure 2- Square Wave Complete Circuit

We connected the 3 MSBs of volume to switches on the DE1 board and set the other bits to a constant '1' level. We also set switches for octave and note number.

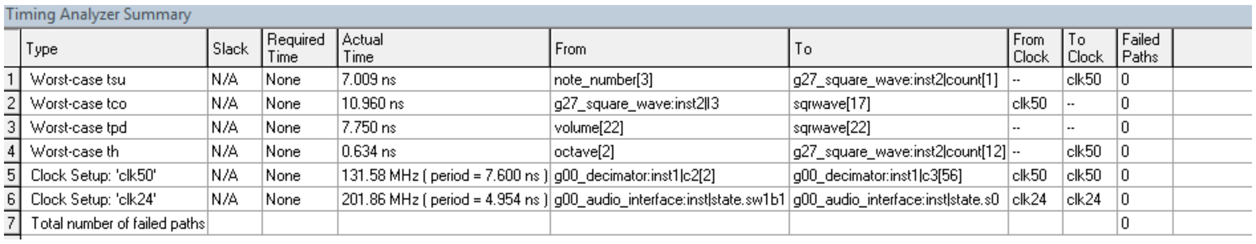
Once the board schematic was compiled and the board programmed, we tested that the notes were correct using an Android music app to check that each note sounded the way it ought to. There are twelve notes in a chromatic scale: C, C#, D, D#, E, F, F#, G, G#, A, A#, B. We compared each note to the true note and confirmed that they sounded the same.

Since there are three bits for octave, we also checked that the notes sounded correct in 2^3 = 8 octaves. 000 results in a very low note, and 111 results in a very high note, as we expect: the number of ones corresponds to the number of bits that the counter input is shifted by, and 8 shifts results in a small period, thus a high frequency, thus a high note, and so on for lower notes and smaller octave inputs.

Volume at first seemed not to work: for volumes higher than 3, the headphones played a scratchy, tinny version of the note rather than a louder version. We think that this might have been an instance of clipping, a phenomenon in which analog output has greater amplitude than a transducer can handle and gets clipped above certain values (not necessarily evenly). We checked with a few other pairs of headphones and found that the volume was adjusted as we expected, so the problem was in the headphones.

Timing Analysis

The circuit seems to have large sequential and combinational logic delays, but each of them is smaller than the clock period (20 ns), and no failed paths were found in the timing summary.



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	7.009 ns	note_number[3]	g27_square_wave:inst2|count[1]	--	clk50	0
2	Worst-case tco	N/A	None	10.960 ns	g27_square_wave:inst2|l3	sqrwave[17]	clk50	--	0
3	Worst-case tpd	N/A	None	7.750 ns	volume[22]	sqrwave[22]	--	--	0
4	Worst-case th	N/A	None	0.634 ns	octave[2]	g27_square_wave:inst2|count[12]	--	clk50	0
5	Clock Setup: 'clk50'	N/A	None	131.58 MHz (period = 7.600 ns)	g00_decimator:inst1|c2[2]	g00_decimator:inst1|c3[56]	clk50	clk50	0
6	Clock Setup: 'clk24'	N/A	None	201.86 MHz (period = 4.954 ns)	g00_audio_interface:inst|state.sw1b1	g00_audio_interface:inst|state.s0	clk24	clk24	0
7	Total number of failed paths								0

Figure 3- Setup, combinational, propagation delay, and hold times

The 7.75 ns propagation delay is visualized in the following simulation waveforms.

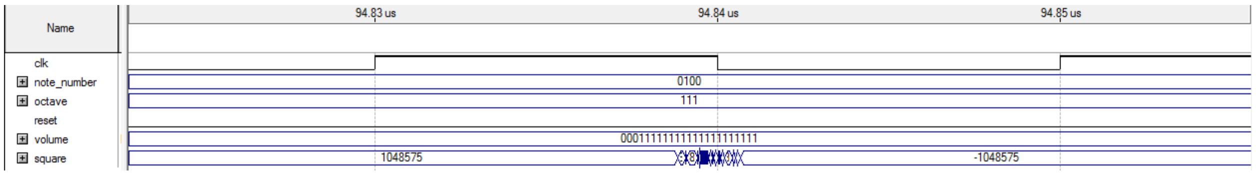


Figure 4- plus to minus

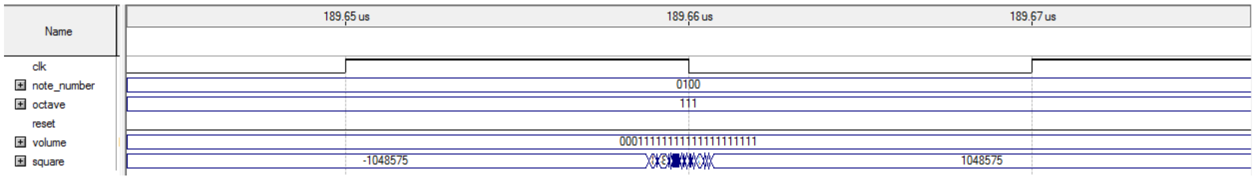
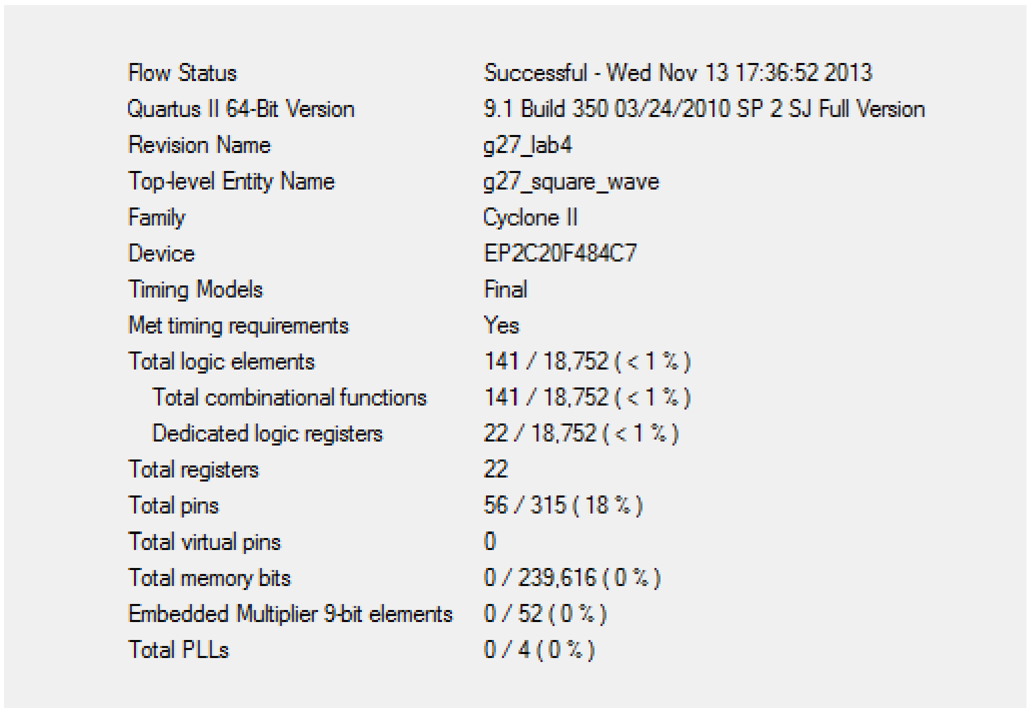


Figure 5- minus to plus

We noticed a glitch when square wave switches from +volume to –volume and vice versa. Since the glitch does not occur near the rising edge of the clock, affects no other part of the circuit's operation, and doesn't affect the sound (it occurs over only 2 ns, and the human ear can only discern sounds of duration > 1/20KHz = 50 us), we consider it harmless.

FPGA Resouce Utilization



Flow Status	Successful - Wed Nov 13 17:36:52 2013
Quartus II 64-Bit Version	9.1 Build 350 03/24/2010 SP 2 SJ Full Version
Revision Name	g27_lab4
Top-level Entity Name	g27_square_wave
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	141 / 18,752 (< 1 %)
Total combinational functions	141 / 18,752 (< 1 %)
Dedicated logic registers	22 / 18,752 (< 1 %)
Total registers	22
Total pins	56 / 315 (18 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 6- Square Wave Generator Alone