Project 1 Report

INTRODUCTION

For my first project utilizing the FPGA Basys2, I wanted to demonstrate the mechanism behind simple technology that perhaps most people take for granted. Every day, no matter where you are you are likely to hear the sirens of a police vehicle or ambulance. The purpose of this project is to explore the hardware and software behind these distinctly recognizable sounds. When a policeman or emergency medical team get a call to action, one of the first things they will do is activate their sirens. Their distinct sound lets every person within ear shot of the vehicles cries that someone needs to get somewhere quick, potentially to save someone’s life. Policeman and emergency medical teams rely on this simple piece of technology to ensure the safety of their own lives as well as others’ while they travel to their destination. The question at hand is as follows: what kind of mechanisms are employed to ensure the reliability of these sirens that every emergency responder vehicle possesses? The answer can be much simpler that what most people might think. This simple technology can be demonstrated using a short script using Verilog and a Basys2 board.

SOFTWARE

The following represents the source code of what causes one of these sirens to activate at a moment’s notice.

module siren(clk, speaker);

input clk;

output speaker;

reg[27:0] tone;

always @ (posedge clk)

tone <= tone+1;

wire[6:0] fastsweep = (tone[22] ? tone[21:15]: ~tone[21:15]);

wire[6:0] slowsweep = (tone[25] ? tone[24:18]: ~tone[24:18]);

wire[14:0] clkdivider = {2’b01, (tone[27] ? slowsweep : fastsweep), 6’b000000};

reg[14:0] counter;

always @ (posedge clk)

if (counter == 0) counter <= clkdivider;

else counter <= counter-1;

reg speaker;

always @ (posedge clk)

if (counter == 0 && switch == 1) speaker <= ~speaker;

endmodule

ANALYSIS

The code above will alternate between two tones. “tone[21:15]” refers to the fast siren, and “tone[24:18]” refers to a slow siren. The “tone” variable is a 28-bit counter which allows the counter to toggle between the fastsweep and the slowsweep about once every six seconds. The slowsweep will gradually increase in pitch with a frequency that is defined by the rate at which the bits in “tone[24:18]” increment, as “tone” is incremented with every positive edge of the clock. The rate at which the slowsweep increases its pitch occurs slower than the fastsweep because as “tone” increments with the positive edge of the clock, it takes a longer amount of time to affect the bit range of slowsweep than that of fastsweep. This is illustrated by the fact that slowsweep is set with a higher range of bits in our 28-bit counter than the fastsweep. In order to achieve the decrease in pitch for slowsweep, we simply invert its bits once “tone[25]” gets set as “tone” increments by 1. The same is seen with fastsweep, where its bits are inverted every time the 23rd bit (read “tone[22]”) gets set. Finally, to switch from slowsweep to fastsweep, we pad the “tone” variable with a few bits. Once the 28th bit of “tone” gets flipped, the clkdivider will switch to either the slowsweep or the fastsweep. It is important to note that without the “switch” variable, this siren would not turn off unless the entire system lost power. But we want the policemen or EMT to be able to switch the siren on and off, so we add a “switch” variable which is mapped to a switch on the Basys2 board. So, the speaker will only output sound when the switch is on. Further, when the siren is on, the speaker only outputs sound with the positive edge of the clock, but because the clock’s frequency is so quick, we perceive the speaker’s output as being continuous.

SOURCES

<http://www.fpga4fun.com/MusicBox2.html> - I used this source for the Verilog source code as well as its explanation of how the siren exhibits its sweeping pitch effect.