ECE271, Final Project

Ben Adams, Grant Haines, Benjiman Walsh

December 1, 2019

Contents

1 Introduction

The purpose of this project is to create a digital logic design that uses an NES controller to control various kinds of output through an FPGA.

2 High Level Descriptions

3 Controller Descriptions

4 HDL Components

5 Appendix

5.1 Source Code

5.1.1 NES Controller Reader

5.1.2 Square Wave Generator

```
module periodTime(input logic clk,
                    input logic [2:0] data,
                    output logic q);

int compareNumber;
int count;

always_comb
        case(data)
            0: compareNumber = 6400;    // just mod input clock until audio spectrum periods
            1: compareNumber = 3200;    // one octave
            2: compareNumber = 1600;
            3: compareNumber = 8000;

            4: compareNumber = 4000;
            5: compareNumber = 2000;
            6: compareNumber = 1000;
            7: compareNumber = 500;        // consider adding default case
        endcase

always_ff @(posedge clk)
```

```
    begin
        if (count >= compareNumber)    // could modify to not restart notes when changed
            count <= 0;
        else
            count <= count +1;
    end

always_comb
    begin
            if( count < compareNumber)
                q = (count > compareNumber/2);     //assigns output with initial low
            else
                q = 0;
    end

endmodule
```

5.2 Simulation Results

5.2.1 NES Controller Reader

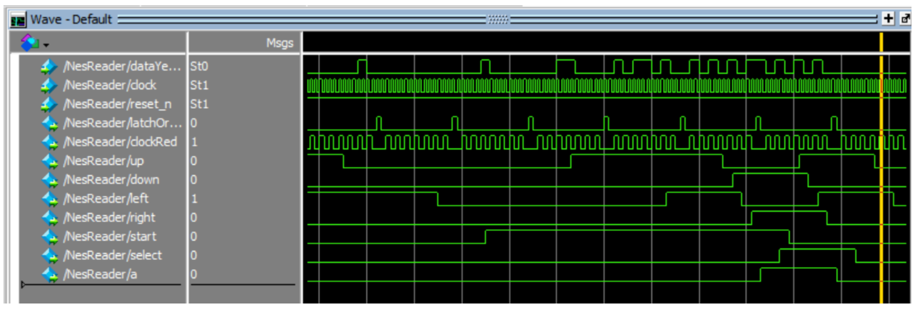

Figure 1: "Button Mashing" on the NES

At first, I wanted to test the NES controller reader by just simulating a bunch of random inputs as seen in Figure 1. I remembered the NES game CONTRA had a cheat code that involved most of the controller's buttons (all but SELECT). The "Contra Code" was then simulated I'm gonna rewrite this

Figure 2: CONTRA screenshot

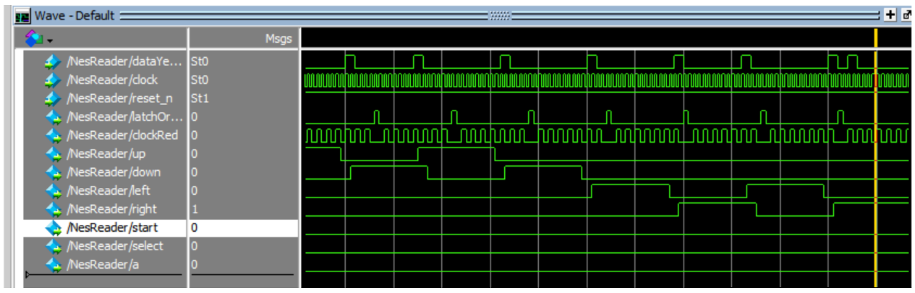

Figure 3: Simulating the "Contra Code"

```
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    1 {80 ps} , 0 {100 ps} , 0 {120 ps} , 0 {140 ps} #up
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 1 {100 ps} , 0 {120 ps} , 0 {140 ps} #down
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    1 {80 ps} , 0 {100 ps} , 0 {120 ps} , 0 {140 ps} #up
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 1 {100 ps} , 0 {120 ps} , 0 {140 ps} #down
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 0 {100 ps} , 1 {120 ps} , 0 {140 ps} #left
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 0 {100 ps} , 0 {120 ps} , 1 {140 ps} #right
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 0 {100 ps} , 1 {120 ps} , 0 {140 ps} #left
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 0 {100 ps} , 0 {120 ps} , 1 {140 ps} #right
force -freeze sim:/NesReader/dataYellow 0 0, 1 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 0 {100 ps} , 0 {120 ps} , 0 {140 ps} #b
force -freeze sim:/NesReader/dataYellow 1 0, 0 {20 ps} , 0 {40 ps} , 0 {60 ps} ,
    0 {80 ps} , 0 {100 ps} , 0 {120 ps} , 0 {140 ps} #a
force -freeze sim:/NesReader/dataYellow 0 0, 0 {20 ps} , 0 {40 ps} , 1 {60 ps} ,
    0 {80 ps} , 0 {100 ps} , 0 {120 ps} , 0 {140 ps} start
```

5.2.2 Square Wave Generator

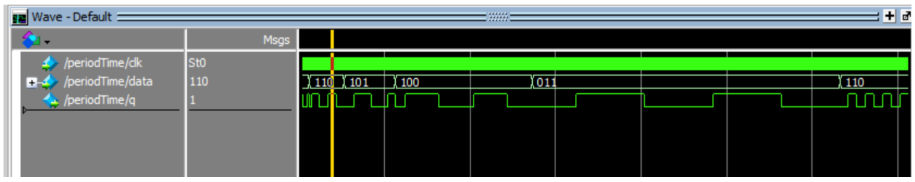

Figure 4: Simulating button inputs to control the square wave oscillator