ECE 271, Example Design Project, Group 0

Matthew Shuman

May 9th, 2018

Contents

This document is a template for the general formatting of the design report. The LaTex source is provided, and will help your team create a high quality professional report. Your team may opt to use another word processor, but it will require substantial effort to produce a high quality report.

1 Project Description

Inputs: This design reads a VCR remote, a PS/2 Keyboard, and/or a 272 Button board. If more than one input source is used for this project, use some of the 4 DIP switches to select which source is enabled at a given time.

Outputs: There is one set of outputs, for the SNES console.

Note that the diagram in figure 1 a good starting point for the top level diagram for the 2018 project.

A good top level diagram would have:

1. Clock oscillator (default 2.08 MHz)
2. A clock divider to make slower speeds
3. Include the input clock for necessary blocks, such as the VCR remote receiver

The hardware diagram in figure 2 is useful for showing the pin connections between the FPGA and hardware modules. Good hardware diagrams have the following items:

1. Power and ground connections for each hardware module
2. Pin numbers being used on the FPGA
3. Descriptions of how the wires are connected, such as wire colors

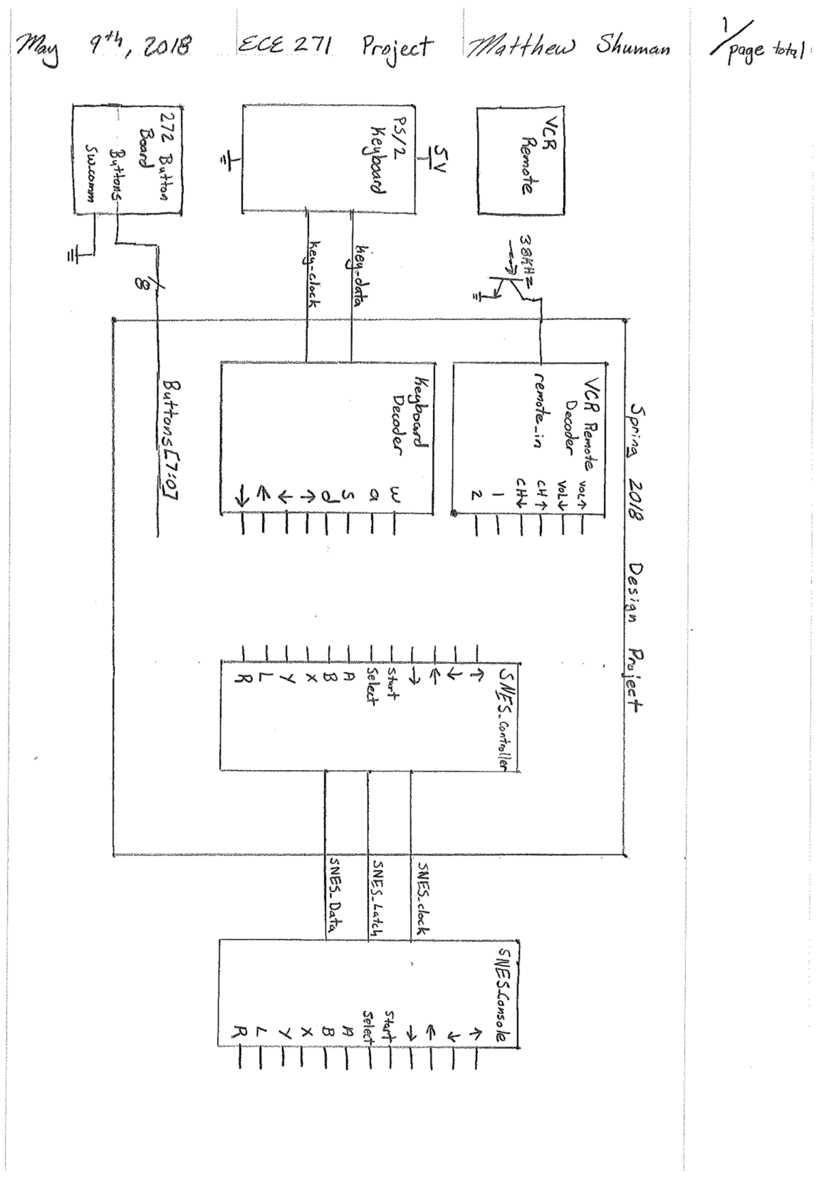

Figure 1: This image is legible, and conveys the point of the design. Your image can be hand drawn, but it must have straight lines, use your OSU ID. I don't recommend drafting this on the computer, because there aren't any decent tools to draw these block diagrams quickly.

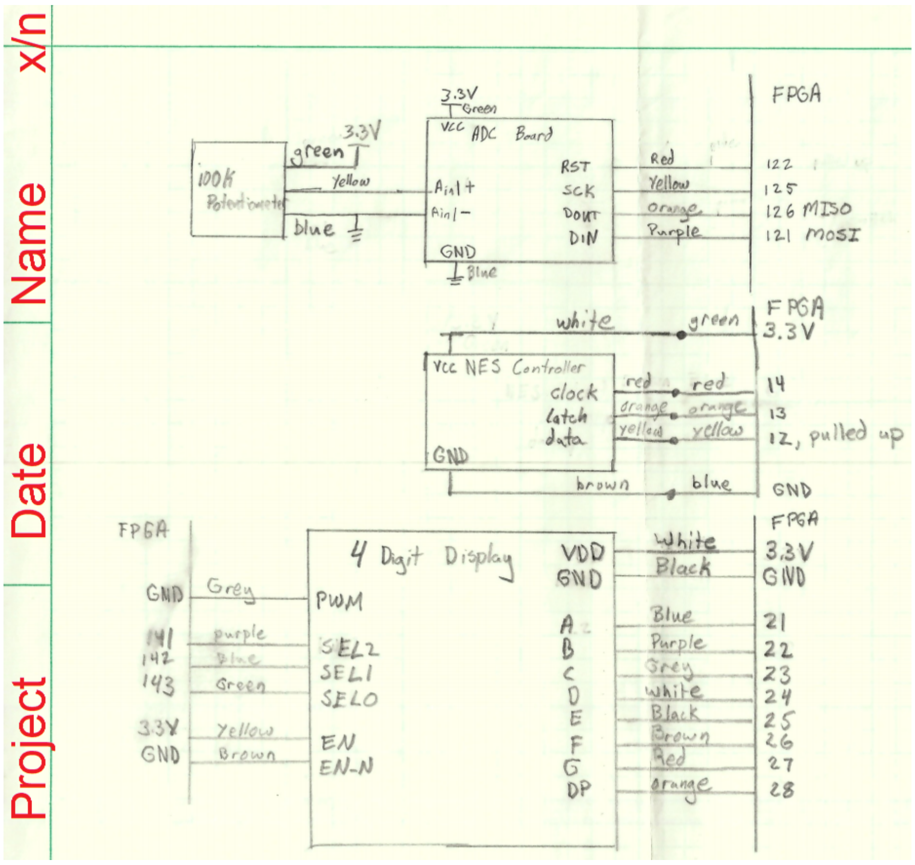

Figure 2: The hardware diagram shows which pins are used on the FPGA, module boards, and relevant supply voltages for the different pieces of hardware used in the system.

2 High Level Description

Inputs: This reads a NES controller and the analog voltage of a potentiometer.
Outputs: This displays a 4 digit value on a seven segment display.

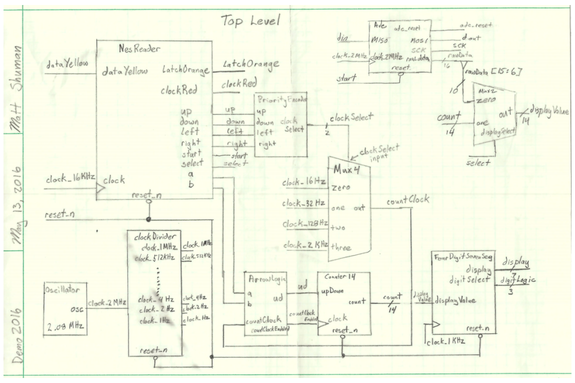

Figure 3: The top level design for the 2016 project. This would be improved by combining the priority encoder and Mux4 into a single clock select block. Combining the ArrowLogic and Counter14 would also make this diagram better. Use chapter 1 concepts wisely on this diagram, specifically hierarchy, modularity, regularity, and discipline.

Put your simulation results for the TopLevel results here

2.1 Functional Unit

Inputs: This reads a 14 bit value, uses a 1 KHz clock signal, and has an active low reset.

Outputs: display[6:0] will control which number is displayed on the seven segment display. A 0 means that segment LED is on and a 1 means that segment LED is off. digitSelect[2:0] controls which digit is illuminated. The table below shows how the digitSelect operates.

000	1's digit
001	10's digit
011	100's digit
100	1000's digit

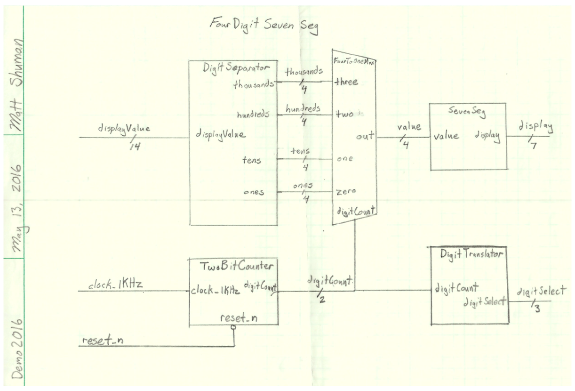

Figure 4: This is an expanded view of the block shown in the high level digram.

Put the simulation results for the functional unit here.

2.1.1 Individual Block

The individual block shown in figure 5 was lab 3 of the ECE 272 Lab.
Inputs: value[3:0] ranges between 0 and 15
Outputs: display[6:0] determines which LEDs should be on to display a number on the seven segment display. A zero turns on the LED, a 1 turns off the LED.

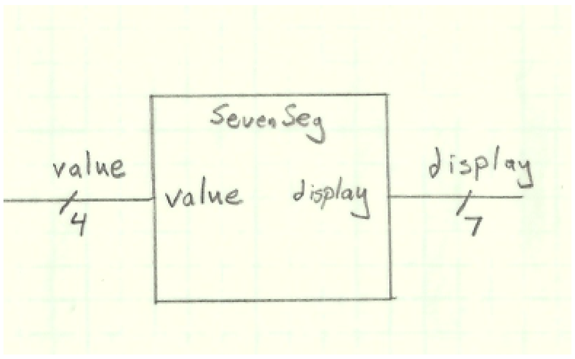

Figure 5: This block was done in lab 3, by making K-Maps and drawing the logic gates needed to make this block of combinational logic. In chapter 4 this was done in System Verilog.

Put the simulation results for the individual blocks here.

2.1.2 Next Individual Block

2.1.3 Next Individual Block

2.1.4 Next Individual Block

2.2 Next Functional Unit

2.2.1 Individual Block

2.2.2 Next Individual Block

2.2.3 Next Individual Block

2.3 Next Functional Unit

2.3.1 Individual Block

2.3.2 Next Individual Block

2.3.3 Next Individual Block

2.3.4 Next Individual Block

2.3.5 Next Individual Block

A SystemVerilog Files

```
//////////////////////////////////////////////////////////////////////////////////
// Company:         Oregon State University
// Engineer:        Matthew Shuman
//
// Create Date:     05/11/2016
// Design Name:     demo2016
// Module Name:     TopLevel
// Project Name:
// Target Devices:  MachX02
// Tool versions:   Lattice Diamond 3.7
// Description:     Top Level for connecting a NES controller to a counter that is displayed on a 4 digit
    display.
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

```

```systemverilog
module TopLevel(
  input logic dataYellow,
  output logic latchOrange, clockRed,
  input logic reset_n,
  output logic [6:0] display,
  output logic [2:0] digitSelect
);

logic up, down, left, right, a, b;
logic ud, countClock;
logic [1:0]clockSelect;
logic clock_16Hz, clock_32Hz, clock_128Hz, clock_1KHz, clock_2KHz, clock_16KHz;
logic [13:0] count;

//This is an instance of a special, built in module that accesses our chip's oscillator
OSCH #("2.08") osc_int (         //"2.08" specifies the operating frequency, 2.08 MHz. Other clock
     frequencies can be found in the MachX02's documentation
  .STDBY(1'b0),                  //Specifies active state
  .OSC(clock_2MHz),                       //Outputs clock signal to 'clk' net
  .SEDSTDBY()                    //Leaves SEDSTDBY pin unconnected
);


//This module is instantiated from another file, 'NesReader.sv'
NesReader reader1(
  .dataYellow(dataYellow),
  .reset_n(reset_n),
  .latchOrange(latchOrange),
  .clockRed(clockRed),
  .up(up),
  .down(down),
  .left(left),
  .right(right),
  .a(a),
  .b(b),
  .clock(clock_32KHz)
);

//This module is instantiated from another file, 'ClockDivider'
ClockDivider Divider1(
  .clock_2MHz(clock_2MHz),       // the clock driving the counter
  .reset_n(reset_n),             // active low reset
  .clock_1MHz(clock_1MHz),       //
  .clock_512KHz(clock_512KHz),   //
  .clock_256KHz(clock_256KHz),   //
  .clock_128KHz(clock_128KHz),   //
  .clock_64KHz(clock_64KHz),     //
  .clock_32KHz(clock_32KHz),     //
  .clock_16KHz(clock_16KHz),     //
  .clock_8KHz(clock_8KHz),       //
  .clock_4KHz(clock_4KHz),       //
  .clock_2KHz(clock_2KHz),       //
  .clock_1KHz(clock_1KHz),       //
  .clock_512Hz(clock_512Hz),     //
  .clock_256Hz(clock_256Hz),     //
  .clock_128Hz(clock_128Hz),     //
  .clock_64Hz(clock_64Hz),       //
  .clock_32Hz(clock_32Hz),       //
  .clock_16Hz(clock_16Hz),       //
  .clock_8Hz(clock_8Hz),         //
  .clock_4Hz(clock_4Hz),         //
  .clock_2Hz(clock_2Hz),         //
  .clock_1Hz(clock_1Hz)          //
);

//This module is instantiated from another file, 'PriorityEncoder.sv'
PriorityEncoder Encoder1(
  .up(up),
  .down(down),
  .left(left),
  .right(right),
  .clockSelect(clockSelect)
);

//This module is instantiated from another file, 'Mux4.sv'
Mux4 Mux1(
  .clockSelect(clockSelect),
  .zero(clock_16Hz),
  .one(clock_32Hz),
  .two(clock_128Hz),
  .three(clock_2KHz),
  .out(countClock)
);

//This module is instantiated from another file, 'ArrowLogic.sv'
ArrowLogic Combo1(
  .a(a),
  .b(b),
  .countClock(countClock),
  .ud(ud),
  .countClockEnabled(countClockEnabled)
);

//This module is instantiated from another file, 'Counter14.sv'
Counter14 Counter1(
  .reset_n(reset_n),
  .upDown(ud),
  .clock(countClockEnabled),
  .count(count)
);

//This module is instantiated from another file, 'FourDigitSevenSeg.sv'
FourDigitSevenSeg(
  .reset_n(reset_n),
  .displayValue(count),
  .clock_1KHz(clock_1KHz),
  .display(display),
  .digitSelect(digitSelect)
);


endmodule
```

A.1 Four Digit Display

```
////////////////////////////////////////////////////////////////////////////////
// Company:          Oregon State University
// Engineer:         Matthew Shuman
//
// Create Date:      05/04/2016
// Design Name:      demo2016
// Module Name:      FourDigitSevenSeg
// Project Name:
// Target Devices:  MachX02
// Tool versions:   Lattice Diamond 3.7
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module FourDigitSevenSeg(
 input logic clock_1KHz, reset_n,
 input logic [13:0] displayValue,        // The value that gets split and displayed on the display,
      number ranges go 0-9999.
 output logic [6:0] display,             // 6543210 -> gfedcba
 output logic [2:0] digitSelect          // 000 -> LSB, 001, 011, 100
 );

logic [3:0] thousands;
logic [3:0] hundreds;
logic [3:0] tens;
logic [3:0] ones;
logic [3:0] value;
logic [1:0] digitCount;

TwoBitCounter i1(
        .clock_1KHz      (clock_1KHz),
        .reset_n         (reset_n),
        .digitCount      (digitCount)
);

DigitTranslator i2(
        .digitCount      (digitCount),
        .digitSelect     (digitSelect)
);

DigitSeparator i3(
        .displayValue    (displayValue),
        .thousands       (thousands),
        .hundreds        (hundreds),
        .tens            (tens),
        .ones            (ones)
);

FourToOneMux i4(
        .thousands       (thousands),
        .hundreds        (hundreds),
        .tens            (tens),
        .ones            (ones),
        .digitCount      (digitCount),
        .value           (value)
);

SevenSeg i5(
        .value           (value),
        .display         (display)
);

endmodule
```

```
////////////////////////////////////////////////////////////////////////////////
// Company:          Oregon State University
// Engineer:         Matthew Shuman
//
// Create Date:      05/09/2016
// Design Name:      demo2016
// Module Name:      DigitSeparator
// Project Name:
// Target Devices:  MachX02
// Tool versions:   Lattice Diamond 3.7
// Description:     A block of combinational logic that will separate a value (0-9999) into digits to be
     displayed.
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module DigitSeparator(
 input logic [13:0] displayValue, //
 output logic [3:0] thousands,      //the MSB digit
 output logic [3:0] hundreds,       //the 100's digit
 output logic [3:0] tens,           //the 10's digit
 output logic [3:0] ones           //the LSB digit
 );

   assign thousands = (displayValue / 1000) % 10; //MSB Display
   assign hundreds = (displayValue / 100) % 10;
   assign tens = (displayValue / 10) % 10;
   assign ones = displayValue % 10;

endmodule
```

```
////////////////////////////////////////////////////////////////////////////////
// Company:          Oregon State University
// Engineer:         Matthew Shuman
//
// Create Date:      05/09/2016
// Design Name:      demo2016
// Module Name:      SevenSeg
// Project Name:
// Target Devices: MachX02
// Tool versions:  Lattice Diamond 3.7
// Description:    A clean SevenSegment display driver
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module SevenSeg(
 input logic [3:0] value,  //the input to display
 output logic [6:0] display// a, b, c, d, e, f, g, the individual LED output for the seven segment along
       with the digital point
 );

  always_comb
    case(value)          //         gfedcba  All LEDs are active low
      4'd0 : display = 7'b1000000; //display 0
      4'd1 : display = 7'b1111001; //display 1
      4'd2 : display = 7'b0100100; //display 2
      4'd3 : display = 7'b0110000; //display 3
      4'd4 : display = 7'b0011001; //display 4
      4'd5 : display = 7'b0010010; //display 5
      4'd6 : display = 7'b0000010; //display 6
      4'd7 : display = 7'b1111000; //display 7
      4'd8 : display = 7'b0000000; //display 8
      4'd9 : display = 7'b0010000; //display 9
      default : display = 7'b0111111; //dash
    endcase
endmodule
```

B Simulation Files (Do scripts)

```
vsim work.DigitSeparator

add wave displayValue
add wave thousands
add wave hundreds
add wave tens
add wave ones

force displayValue 10#1234 0
force displayValue 10#5678 10

run 100 ps
```