LED Wall
By Evan Dagg

The goal of the project was to make a 60 x 30 RGB LED display. This consisted of 30 strands stacked vertically with 60 RGB LEDs on each.
Materials

• FPGA – Machx02-7000HE
  – Lattice Diamond Software
• Microcontroller – Teensy 2.0 (ATmega32U4)
• LED – WS2812B RGB LED
LED – WS2812B RGB LED

- Flexible PCB strand of 60 LEDs/meter
- LED is RGB
- +5V, GND, DIN/DO
LED – WS2812S RGB LED

Power

• Max single LED current = 60mA
  – Full brightness, white
• 60 LED per strand – max strand current = 3.6A
• 1800 LEDS total – max current = 108A @ 5V
LED – WS2812S RGB LED

- 24 bits of data/LED (1 byte per R, G, B)
- 256 levels of brightness for each color

### Composition of 24bit data:

<table>
<thead>
<tr>
<th>G7</th>
<th>G6</th>
<th>G5</th>
<th>G4</th>
<th>G3</th>
<th>G2</th>
<th>G1</th>
<th>G0</th>
<th>R7</th>
<th>R6</th>
<th>R5</th>
<th>R4</th>
<th>R3</th>
<th>R2</th>
<th>R1</th>
<th>R0</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
</table>

Note: Follow the order of GRB to sent data and the high bit sent at first.

### Data transfer time (TH+TL=1.25μs±600ns)

<table>
<thead>
<tr>
<th></th>
<th>T0H</th>
<th>1 code high voltage time</th>
<th>0.4us</th>
<th>±150ns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T1H</td>
<td>1 code high voltage time</td>
<td>0.8us</td>
<td>±150ns</td>
</tr>
<tr>
<td></td>
<td>T0L</td>
<td>0 code low voltage time</td>
<td>0.85us</td>
<td>±150ns</td>
</tr>
<tr>
<td></td>
<td>T1L</td>
<td>1 code low voltage time</td>
<td>0.45us</td>
<td>±150ns</td>
</tr>
<tr>
<td></td>
<td>RES</td>
<td>low voltage time</td>
<td>Above 50μs</td>
<td></td>
</tr>
</tbody>
</table>
LED – WS2812S RGB LED

• Data is cascaded through LEDs
LED – WS2812S RGB LED

• Testing on a microcontroller
  – Example of LED strands working

• Much easier to test on microcontroller vs. FPGA

• Now to implement on the FPGA
  – With an FPGA you can update each strand at the same time in parallel, a microcontroller would update in series taking 30x longer
Inputs
- SCK
- SSEL
- MOSI
- switch
- reset_n

Outputs
- LED_out_0
- LED_out_1
- ...
- LED_out_28
- LED_out_29
FPGA – Machx02-7000HE
LED_OUT_SM part 1

```vhdl
always @ (posedge clk, negedge reset_n)
begin
  if (!reset_n)
    begin
      state <= my_reset;
      nextbit <= 0;
      led_out <= 0;
    end
  else
    begin
      case (state)
        my_idle : begin
          led_out <= 1;
          nextbit <= 0;
          if ( (count_LED == 16) && (bitin) ) //if bitin = 1 count to 16 (.8uS) and switch states
            state <= my_one;
          else if ( (count_LED -- 0) && (!bitin) ) //if bitin = 0 count to 0 (.4uS) and switch states
            state <= my_zero;
          else if ( !reset ) //reset signal from buffer_ctrl produces a low signal at output
            begin
              led_out <= 0;
              state <= my_reset;
            end
        end
        my_zero : begin //bitin = 0
          led_out <= 0; //set LED out to low
          if (count_LED == 24) //stay low for a count of 16 (.8SuS) totaling a count of 24
            begin
              state <= my_reset;
              nextbit <= 1; //ready for nextbit
            end
        end
      endcase
    end
end
```
my_one : begin
    // bitin = 1
    led_out <= 0;  // set LED_out to low
    if (count_LED == 24)  // stay low for a count of 16(.85μS) totaling a count of 24
        begin
            state <= my_reset;
            nextbit <= 1;  // ready for nextbit
        end
    end
my_reset: begin
    nextbit <= 0;
    if (ws2812_reset)
        led_out <= 0;
    else if (shifting)  // once shifting happens I have new data and reset is over
        state <= my_idle;
    end
default: begin
    state <= my_reset;
    led_out <= 0;
end
endcase
end
FPGA – Machx02-7000HE

• Issue
  – FPGA SPI module takes bytes, stores to RAM, increments RAM WrAddress, and repeats
    • This effectively writes to one row before moving on to the next column
  – But FPGA updates each column at the same time, moves to next column, and repeats.
  – Need to accommodate for this in buffer_ctrl
FPGA – Machx02-7000HE

buffer_ctrl

```verilog
assign RdAddress = row_count + column_addr;
always @(posedge clk, negedge reset_n)
    begin
        .
        .
        .
        if (row_count != 5760)
            begin
                row_count <= row_count + 180;
            end
        else if (buffer_empty)
            begin
                column_addr <= column_addr + 1;
                row_count <= 0;
            end
        end
```
Microcontroller – Teensy 2.0 (ATmega32U4)

• To demo it we needed something to be displayed
• 5400 byte array
  – 3 bytes = 1 LED
  – 0x8000FF = 50% green, 0% red, 100% blue
  – Update array, transmit over SPI
Microcontroller – Teensy 2.0 (ATmega32U4)

• SPI Transmit Array – 1800 byte array

```c
void transmit (void)
{
    int i;
    for (i = 0; i < 1800; i++)
    {
        switch (array[i])
        {
            case 0: SPI_masterTransmit(0xCC); SPI_masterTransmit(0xCC); SPI_masterTransmit(0xCC); break; //no LED on
            case 1: SPI_masterTransmit(0xFF); SPI_masterTransmit(0xCC); SPI_masterTransmit(0xCC); break; //green LED
            case 2: SPI_masterTransmit(0xCC); SPI_masterTransmit(0xFF); SPI_masterTransmit(0xCC); break; //red LED
            case 3: SPI_masterTransmit(0xCC); SPI_masterTransmit(0xCC); SPI_masterTransmit(0xFF); break; //blue LED
            default: SPI_masterTransmit(0xCC); SPI_masterTransmit(0xCC); SPI_masterTransmit(0xCC); break; //default turns off LEDs
        }
    }
}
```
Microcontroller – Teensy 2.0 (ATmega32U4)

Example - 1

```c
const unsigned char ascii [] = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // space
    0x00, 0x00, 0xBE, 0x00, 0x00, 0x00, // !
    0x7C, 0xA2, 0x92, 0x8A, 0x7C, 0x00, // 0
    0x84, 0x82, 0xFE, 0x80, 0x80, 0x00, // 1
    0x84, 0xC2, 0xA2, 0x92, 0x8C, 0x00, // 2
    0x44, 0x92, 0x92, 0x92, 0x6C, 0x00, // 3
    0x30, 0x28, 0x24, 0xFE, 0x20, 0x00, // 4
    0x9E, 0x92, 0x92, 0x92, 0x62, 0x00, // 5
    0x78, 0x94, 0x92, 0x92, 0x60, 0x00, // 6
    0x02, 0xE2, 0x12, 0x0A, 0x06, 0x00, // 7
    0x6C, 0x92, 0x92, 0x92, 0x6C, 0x00, // 8
    0x0C, 0x92, 0x92, 0x52, 0x3C, 0x00, // 9
};
```
Transmit A Char

Example - 1

0x84 0x82 0xFE 0x80 0x80 0x00

0 0 0 0 0 0
0 1 1 0 0
1 0 1 0 0 0
0 0 1 0 0 0
0 0 1 0 0 0
0 0 1 0 0 0
1 1 1 1 1 1 0

0void transmit_char ( char letter, int p)
1
0{ int number = 0, n, i;
0 block = p/10;
0 if (letter == ' ')
0 number = 0;
0 else if ( letter <= 57) //Numbers
0 number = letter - 46;
0 else if ( letter <= 90) //Capital
0 number = letter - 53;
0
0 for ( i = 0; i < 7; i++) //7 corresponding to 7 rows
0 {
0 for (n = 0; n < 6; n++) //6 corresponding to each column
0 {
0 if ( (ascii[number*6+n] & (0b00000010 << i )) != 0 )
0 {
0 array[ (block*540) + (i*60) + (59 - p*6 - n) ] = 1;
0 }
0 }
0 }
0 }
Demo

• **Snakes on A Wall**

• Issues
  – Floating Ground?
  – Ground Loop?

• Questions?