ECE 212 – Digital Circuits II

DRAFT Overview of the Adafruit RGB LED Matrix

Revised February 1, 2019

1. Overview

Large RGB LED matrices are used to construct video walls and electronic billboards. When viewed from a distance, each LED forms a pixel in a larger video image. Adafruit sells small RGB LED matrices that can be combined to create larger signs. We will work with Adafruit's 16x32 RGB LED matrix, which is shown in Figure 1. This unit contains a total of 512 RBG LEDs organized into 16 rows and 32 columns (in landscape orientation). The array is controlled using 12 pins (6 data signals and 6 control signals), as shown in Figure 1) that can be used to set the color displayed by each LED for a short period of time. To make it appear that the entire display is on, the controlling system must rapidly switch between driving each of the LEDs in the array.

Figure 1 – Adafruit 16x32 RGB LED Matrix

2. System Organization

Since each pixel in the RGB LED matrix requires three signals, the total number of signals required to drive the 512 pixels in the matrix would be 3 x 512 = 1,536 signals. is clearly not practical. Moreover, turning on all 512 LEDs at the same time would consume a prohibitive amount of current.

To address these concerns the LED matrix uses a *time-multiplexing* approach in which only a subset of pixels are turned on at any given time. You may recall that in ECE 211 we designed a seven-segment controller in which each of eight seven-segment digits was enabled one-at-a time in a repeating pattern that made all digits appear "on" to the human eye. The LED matrix uses a similar approach, but instead of enabling individual digits for display it enables two individual *rows* of 32 pixels in a repeating pattern.

Figure 2 shows the organization of the LED driving logic in a panel. The array of 16 x 32 pixels is divided into two half panels (top and bottom). The LEDs in each half panel are organized as 8 *rows* of 32 *columns*.

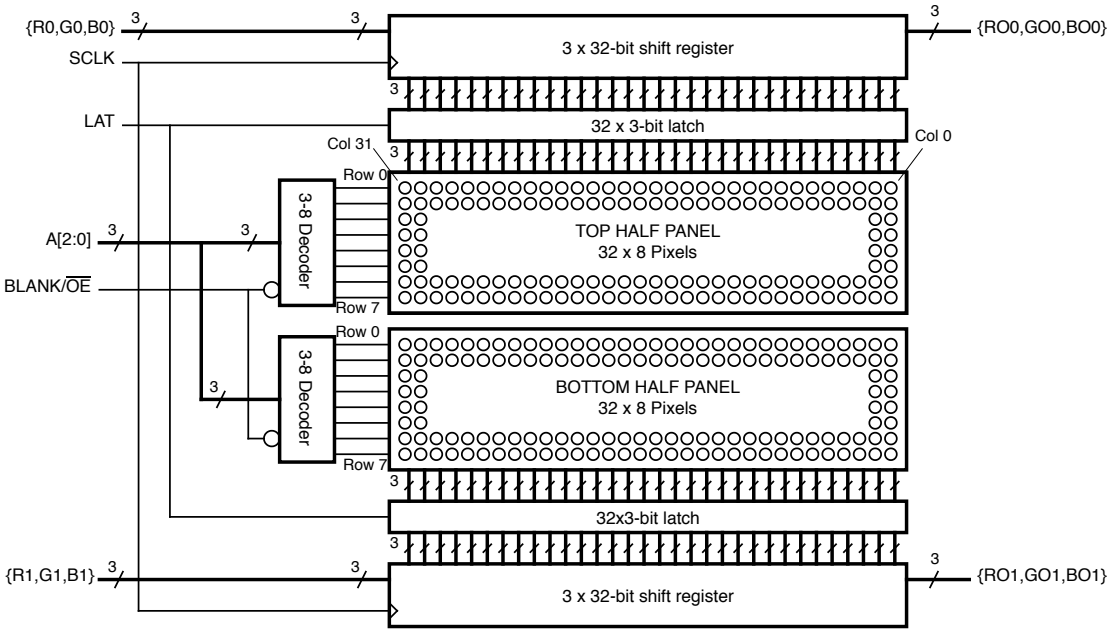

Figure 2 – Panel Organization

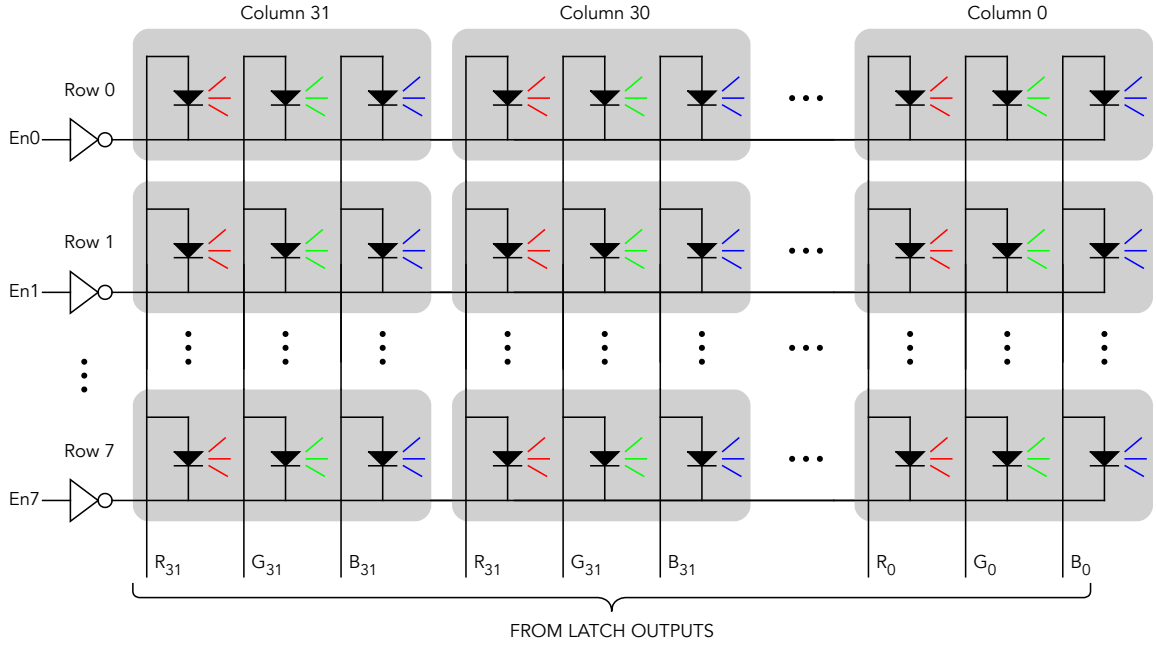

Figure 3 – Pixel (LED) Connection Detail for a Half Panel

The individual LEDs in each pixel are connected in a manner similar to the circuit shown in Figure 3. The *anodes* of the red, green, and blue pixels in each *column* are connected together and driven by the outputs of 32 3-bit latches, while the *cathodes* of all of the pixels in each *row* are connected together and driven by an enable signal (En0, En1, …, En7). Therefore individual pixel will light up only the row enable for that pixel is asserted true while at least one of the three red, green, and blue column connections are also asserted true. If more than one of the column connections is asserted true, then the

pixel will display a blend of the three primary colors so that a total of eight colors can be displayed: black (off), red, green, blue, cyan, magenta, yellow, and white.

Returning to Figure 2, we see that the inputs to the latch in each half panel are connected to the outputs of a 3-bit wide 32-stage shift register along with a latch enable input `LAT`. The shift register is driven by three data inputs `R1`, `B1`, and `G1` (`R2`, `B2`, and `G2` for the bottom half panel) and a clock signal named `SCLK`. The purpose of the shift register is to allow the pixel values for a row to be entered one pixel at a time. Once an entire row has been entered, the `LAT` input can be asserted to store the entire row in the latch, after which the row enable input `{C, B, A}` can be set to enable the display of the appropriate row.

You may also notice that there are six outputs of this circuit `RO1`, `BO1`, `GO1`, `RO2`, `BO2`, and `GO2`. Each of these is connected to the rightmost stage of the shift register, and allows multiple LED matrix units to be connected horizontally in a "daisy chain" to create wider displays that are controlled on a row-by-row basis. For example, connecting two units would allow the creation of a 16 x 64 pixel display.

3. Connecting the LED Matrix

The LED Matrix unit is connected via three connectors, as shown in Figure 4.

3.1 Input Connector

Control and data signals are passed to the LED matrix using the ribbon-cable connector marked "INPUT". We will connect FPGA board outputs to this connector to control the LED matrix.

As shown in 5 (a) the input connector includes the following pins.

- `R1`, `G1`, `B1` – the red, green and blue input values for the top half-panel (in some documents, these are labeled `R0`, `B0`, `G0`).
- `R2`, `G2`, `B2` – the red, green and blue input values for the bottom half-panel (in some documents, these are labeled `R1`, `B1`, `G1`).
- `C`, `B`, `A` – the row-select signal for both the top and bottom half-panels. The quantity `{C, B, A}` selects one row for display in each half-panel. For example, if `{C, B, A} = 3'b101` then row 5 of the top and bottom half-panel will display the current contents of the row latch.
- `SCLK` – the clock signal that drives the shift registers in both half-panels. A rising edge on this signal shifts the current values of `{C, B, A}` into the 32-stage shift register. After 32 clock edges, and entire row is shifted into the shift register.
- `BLANK` / $\overline{\text{OE}}$ (output enable) – When `BLANK` is asserted true all pixels in the panel are turned off. This can also be considered an active-low output enable signal.
- `GND` – the connector for the control and data signals includes four ground pins. *All four of these pins should be connected to ground to ensure proper operation.*

Note: Changing the row select `{C, B, A}` immediately changes which row is being displayed based on the *current* latch contents. This means that they should be changed only when the display is disabled using the `BLANK` input to avoid any potential "ghosting" of colors.

The LED matrix units in AEC 419 are provided with pre-wired cables for connection to the FPGA board including color-coded wires which will be connected to the Nexys4 board using the JA and JB PMOD connectors. Because these units and the FPGA boards will be shared between lab sections, you will use a standard connection scheme for connecting the FPGA to pins on the PMOD connector as shown in Figure 6.

3.2 Power Connector

Power is fed to the LED matrix using a Molex (white plastic) connector in the middle of the board. This connector must be attached to a separate 5V power supply. There are two reasons for using a separate power supply from the FPGA board. First, the Nexys4 FPGA board uses a 3.3V power supply, while the LED matrix requires a 5V power supply. Second, because the LED matrix is lighting up to 64 LEDs simultaneously, it can draw significant current (up to 2A) which is beyond the current driving capabilities of the FPGA board. This also allows the connection of 3.3V control and data signals from the FPGA to the LED matrix.

Note: When selecting a power supply, be aware that if all LEDs are turned on to white at full brightness the LED matrix can draw in excess of 2A. Not all power supplies can provide this, particularly the small "wall warts". For this reason, your control circuit should avoid turning on a large number of LEDs simultaneously.

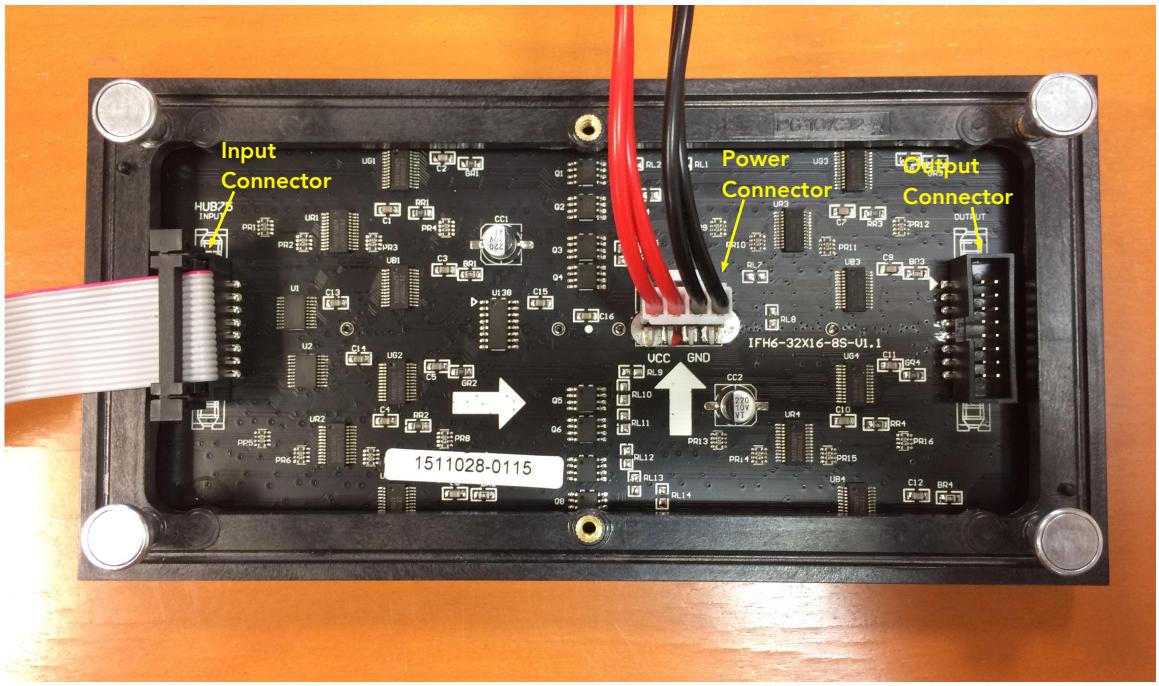

Figure 4 – LED Matrix Connections

(a) Input Connector (b) Output Connector

Figure 5 – Ribbon-Cable Connectors

3.3 Output Connector

As discussed earlier, LED matrix units can be connected together in "daisy-chain" fashion to allow the construction of wider displays. The output ribbon cable connector allows this to be done using a ribbon cable that passes through the row select, SCLK, LAT, and BLANK signals while providing the outputs of the shift register of the LED matrix on the left to the LED matrix on the right. Note this creates a logical LED matrix that is 16 rows by the total number of columns in the connected units. For example, if we connected together two LED matrices then it would appear logically as a 16 x 64 array with 64-bit shift registers. The controller design would need to be modified to accommodate the longer rows.

The pinout for the output connector is shown in Figure 5 (b). Note that its pinout differs from the pinout of the input connector so that the signals can feed through the ribbon cable from shift register outputs to shift register inputs. An important aspect of this different pinout is that *these connectors are not interchangeable. Be sure to connect our control circuit only to the input connector.*

4. Controlling the LED Matrix

4.1 Basic Control Sequence

The following sequence is used to control the LED matrix and allow each LED to display one of the 8 basic RGB colors::

1. Shift in the RGB values for columns 0-31 of row 0 for the top and bottom half-panels using `{R1, G1, B1}` and `{R2, G2, B2}` respectively.
   (a) Set `SCLK` to 0 (if not already) and set the RGB pins to the desired value.
   (b) Set the `SCLK` signal to 1 to shift in one pixel of data.
   (c) Repeat steps (a) and (b) 31 more times to shift in values for all 32 pixels in the row.
2. Disable the display be asserting `BLANK=1`.
3. Update the row select signal `{C, B, A}` to the row corresponding to the pixel values that were just shifted in in Step 1.
4. Assert `LAT=1` to latch the contents of the shift register into the latch and drive the column signals to the LEDs in each column.
5. Assert `BLANK=0` to enable the display of the new row.
6. Wait for some fixed amount of time. The exact time to wait is flexible, but should be long enough for the LEDs to fully turn on while being short enough for the

matrix to be refreshed a sufficient number of times a second so that there is no noticeable flicker..

7. Go back to step 1 and repeat the process of entering pixels for row 1 of the two half panels, followed by rows 2-7 before repeating with row zero.

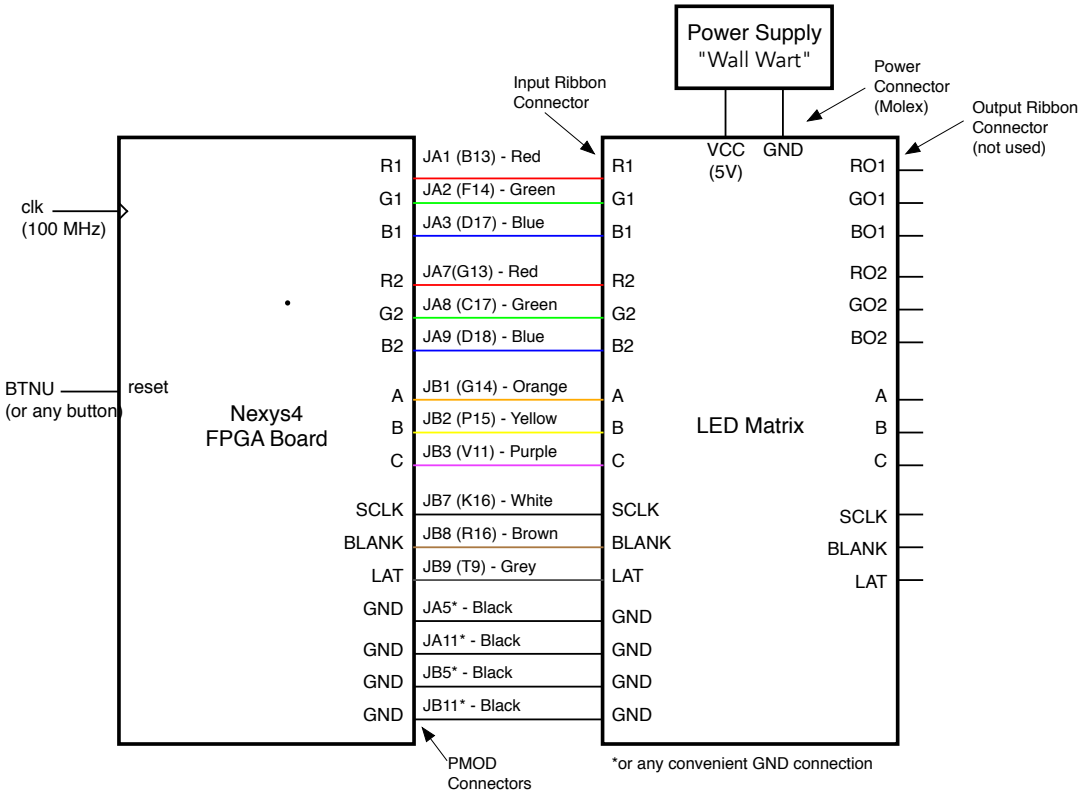

Figure 6 – Connecting the Nexys4 FPGA Board to the LED Matrix

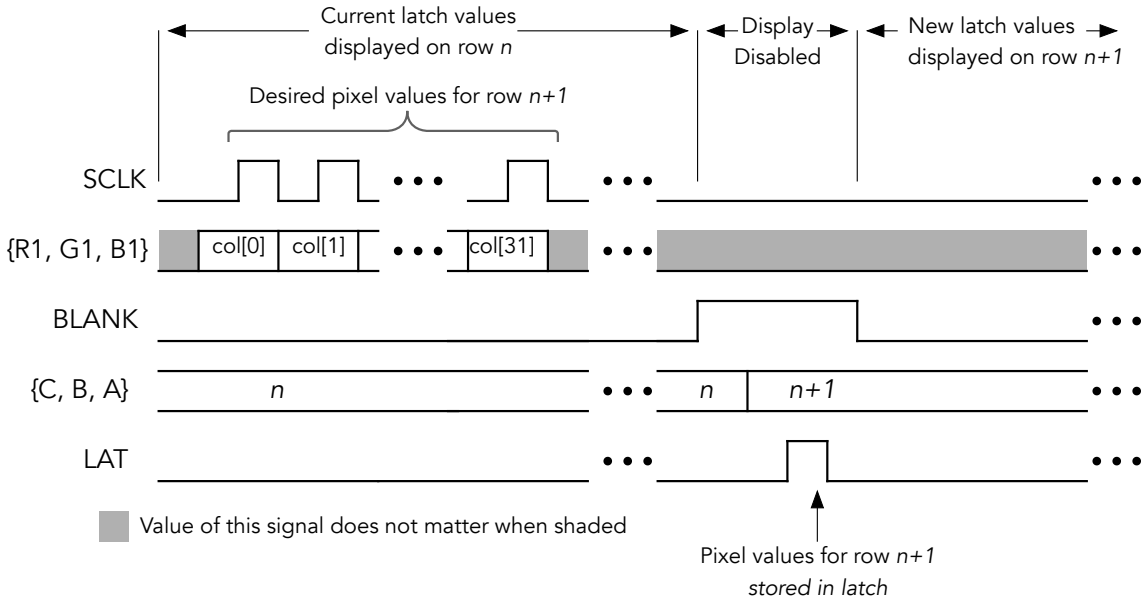

Figure 7 – Example Waveform to Display a Row of the Top Half-Panel

4.2 Enhancing the Control Sequence: More Colors

The sequence described above allows each pixel to display one of 8 basic combinations of red, green, and blue. This provides 3 bits worth of color. To display a wider range of colors, more bits must be used to vary the duty cycle of each color. For example, suppose we would like to represent colors using 12 bits (4 bits per color). We could repeat steps 1-5 of the procedure above with the following modification: for the first iteration, look at the most significant bit to determine which colors to turn on. For the second iteration, look at the next most significant bit and wait half as long as the first iteration in step 5, etc. In this way, the intensity of red, green, and blue in a particular LED can take on a variety of perceived brightness, providing a wider variety of colors and intensities.

When adding more bits it is important to remember that the display still needs to achieve an overall refresh rate of at least 100 Hz to provide a flicker-free appearance. The maximum bit depth can be limited both by the performance of the FPGA driving the matrix and the response rate of the LEDs in the matrix.

References

"Adafruit Documentation": https://learn.adafruit.com/32x16-32x32-rgb-led-matrix/

"RBG LED Panel Tutorial": http://bikerglen.com/projects/lighting/led-panel-1up/