ECE3829: Oscilloscope
A-Term 2020

Prudence Lam

October 14, 2020

1 Introduction

The purpose of the lab was to navigate the VGA display using the VGA controller, the seven segment display, and the light sensor SPI modules from previous labs. The program implements three new features: an oscilloscope, a text display, and a sample count. The oscilloscope is composed of a blue border, 256-pixels long and 512-pixels wide. A moving pixel travels across the scope display at a rate of one second, and has an amplitude that varied depending on the value of the light sensor. My name is displayed in text next to the oscilloscope, with an opacity that changes according to the light level. The sample count is displayed on two of the seven segment displays, and increases whenever a new reading is obtained.

2 Discussion and Results

The final product is a combination of the modules created in previous labs, along with additional sequential and combinational circuits. Using the MMCM, both a 25MHz and a 10MHz internal clock were derived from the FPGA's 100 MHz clock. The 25MHz clock was used for the VGA display, and the 10MHz clock was used for the light sensor SPI and the seven segment modules. The following block diagram summarizes the inputs and outputs to the system from a top level.

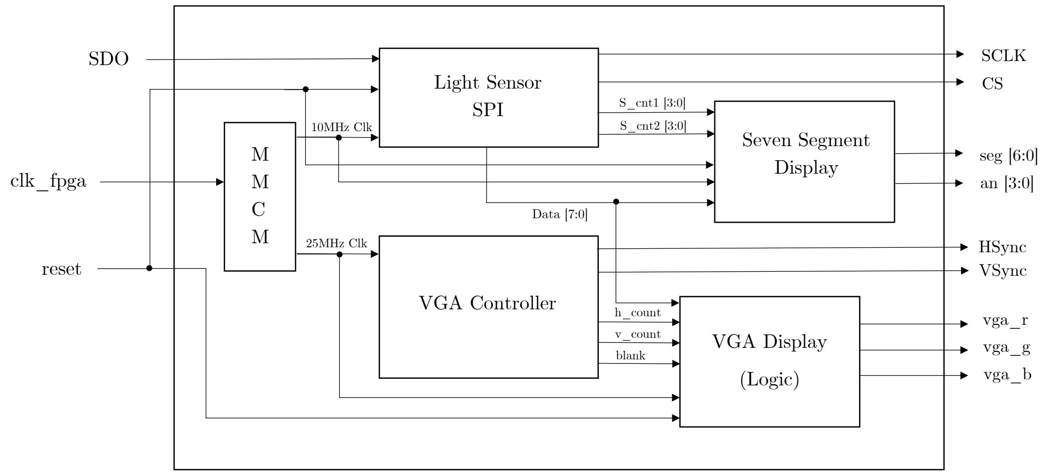

Figure 1: Block diagram of final implementation consisting of the MMCM, the light sensor SPI, the Digilent VGA controller, the logic for the VGA, and the seven segment display.

Each new feature is explained in further detail in **System Requirements**.

2.1 System Requirements

Three new features were implemented alongside the modules created from previous labs.

1. **Oscilloscope**

   The borders of the oscilloscope (512x256 pixels) were first defined using h_count and v_count. They were evaluated against the border's designated position on-screen (511 and 255 respectively, as indexing begins from zero), with relational operators. To create the moving pixel, a counter that counts 48828 cycles was used to down clock the 25MHz clock to 512Hz. Every time the maximum count is reached, a signal initiates another counter to count the number of pixels required, 512. This count is stored in a register and evaluated for equality with `h_count`. The result is a pixel that moves at a rate of 1Hz per pixel, eventually traversing the entire 512 pixel length in 1 second. Similarly, the vertical movement of the pixel is defined by evaluating `v_count` for equality with the difference between 255 and the value from the light sensor. The reason for this is to allow the pixel to travel to the top of the window at a high light sensor reading (FF), and vice versa. Recall that both `h_count` and `v_count` begin counting at the top left of the VGA display.

2. **Sample Count**

   Two additional counters were created in the light sensor SPI module. They determined the output to the seven segment displays showing the sample count. Defined as `seg1` and `seg2`, with the former representing the count's ones place and the latter representing the tens place, they were initiated every time the FSM controlling the light sensor logic reached its display state. This state ensures that the shift register has finished shifting and the data output contains all the necessary bits. Since the tens place depends on the ones place, the counter for `seg2` is placed inside the counter for `seg1`. Each time the `seg2` counter finishes its count from 0 to 9, the other counter increments by 1. A limitation of this is that the sample count cannot go beyond a count of 99. Once it reaches this point, it loops back to 0.

3. **Text Display**

   To display text, a ROM and a multiplexer were used. Using the design shown in class for creating an 8x16 pixel letter (see **Fig 2**), the characters "P, R, U, D, E, N, C" were designed row by row in binary. This was placed inside a case statement with values accessed depending on `v_count` as it counts down the screen.

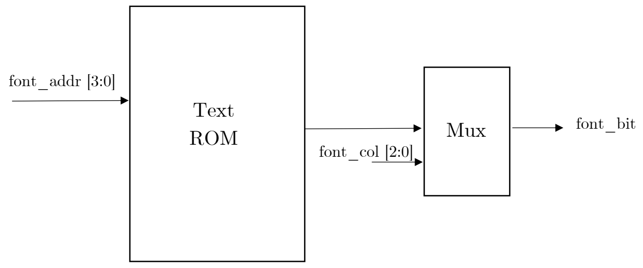

Figure 2: Block diagram of a single 8x16 font, described in class.

The latter half of the code consists of multiple wire assignments to access, store, and combine the positional values of the bits that make up each character. Although my full name is Prudence, the letter "E" is repeated twice. As a result, to display my "E" character, the data was set to the "and" between the set positions of both the "E" characters, and with a variable called `e_bit`. This variable contains the value of a bit in a character row, accomplished by extracting the index of the inverse of `font_col` (inverse due to the position of the MSB and LSB), of the row. A similar procedure is carried out for all the other characters, except only one position is needed as only the characters are unique.

Finally, to display them all at once, an "or" statement was made between the data for each character.

4. **Extra Credit: Changing Opacity**

The color of the text display ranges from white when the value of the light sensor is "FF", black when it is "00", and varying shades of grey for values in-between. This was accomplished by creating a separate register that stores the first four bits of the light sensor data three times in a 12-bit bus. However, a limitation to this is that the shade of grey cannot be fully controlled.

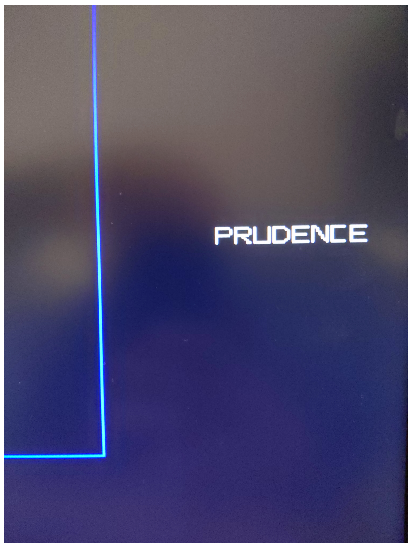

Figure 3: Text display of my name at a light sensor value of "FF". Each character has a dimension of 8x16.

3 Summary and Conclusion

This lab was a culmination of all the previous labs, combining the seven segment, the VGA controller, and the light sensor modules into a functioning program with additional features. The greatest difficulty faced in the design process was in understanding how the text display example worked in class, and extending that to several different characters. I also encountered some difficulty in displaying all of the text on the display from the different parameters that exist, but the test bench was extremely helpful in debugging. Overall, this lab provided a great sense of satisfaction as I learned to successfully interface the different modules I built with each other.

For the future, I would like to expand on the features of the oscilloscope, creating a concrete sine wave instead of a single moving pixel.