OLED Project

Lei(Raymond) Chi
Albert Nerken School of Engineering
The Cooper Union for the Advancement of Science and Art
New York, USA
lei.chi@cooper.edu

I. Introduction

The OLED Project represents a comprehensive exploration into the capabilities of the OLED display integrated into the ZedBoard, with a primary focus on leveraging the AXI bus for efficient communication. This project was undertaken within the framework of the ECE 311 Hardware Design course, providing a valuable opportunity for hands-on experience with Xilinx Vivado and the ZedBoard platform. The project was followed through Vipin Kizheppatt's youtube video where he explains indepth and walh through how to display characters on the ZedBoard display. This venture delves into the practical implementation of hardware design concepts, particularly in the realms of AXI bus integration and OLED display functionalities.

II. Background

A. The OLED interface

The OLED display part number on the Zedboard: UG-2832HSWEG04, with the part number users could search up the datasheet of the OLED display. The OLED display has a total display area of 128(Columns) x 32(Rows) pixels. The display is divided into 4 pages and each page contains 8 rows. In the datasheet, the rows are referred as commons and columns are referred as segments.

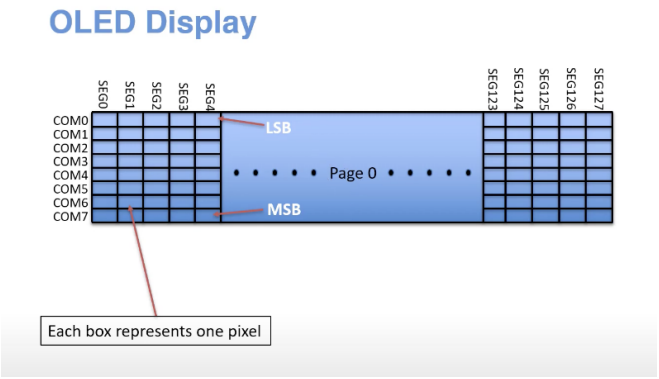

Fig. 1. pixel breakdown of each page on the OLED interface

Figure 1 is a visual representation of the name and the pixels placement of a page on the OLED interface. The user should also be aware of the LSB to MSB direction for the display.

The display contains a 32 x 128 RAM inside of the device. The RAM is called Graphic Display Data RAM(GDRAM) and each location of RAM is correlated to each pixel. The display works by writing into the RAM from outside and the controller would read the RAM, which then display it. The pixel value will present white when it's high and black when it's low.

When interacting with OLED the main interface is SPI interface. The users would send the signal through the SPI interface. There is another signal interacting with the display which is the D/C#, which is the signal that specifies whether the data on SPI interface is a command or a data to the GDRAM. The signal of D/C# would be high if storing a data to GDRAM, and low when it is a command.

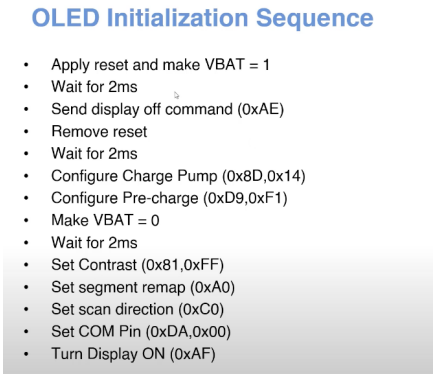

Fig. 2. OLED Initialization Sequence

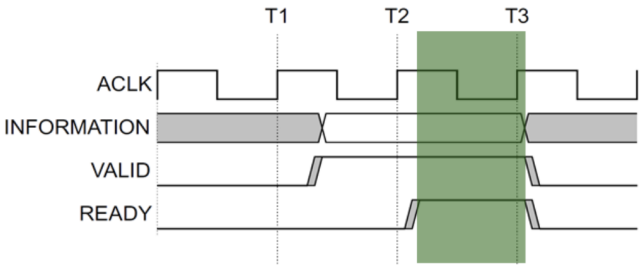

Fig. 3. waveform when data is transmitted

Figure 2 is the sequence that the OLED needs to undergo before the users start interacting with the display.

B. AXI and overall desgin

AXI is short for Advanced eXtensible Interface. AXI is an interface protocol created by ARM from AMBA(Advanced Microcontroller Bus Architecture) standard. AXI contains five channels: read address, read data, write address, write data and write response. These channels run between AXI transmitter and AXI receiver(inclusive terminology).

In Figure 3, the data would be transmitted in AXI protocol when both VALID and READY signals are high, which in this figure is T3. In order to complete a valid read transaction from transmitter to receiver, the read address channel would first send the signal to receiver to set the address and control signals. Secondly, the address that is set would then send from the receiver from the read data channel. On the other hand, to have a valid AXI write transaction, the write address channel would do the same setting the address and control signals to the receiver. Secondly, the write data channel would be activated and set write data signals from the transmitters to the receivers. Lastly, the receiver would send a write response signal back to the transmitter to verify the data have been wrote in the receiver.

III. Methodology

As previously mentioned, the project closely followed Vipin Kizheppatt's YouTube video, meticulously documenting each step executed to display the message on the OLED. Figure 4 showcases the code for oledControl.v, containing snippets that initialize the OLED. (Refer to Figure 2 for a detailed explanation

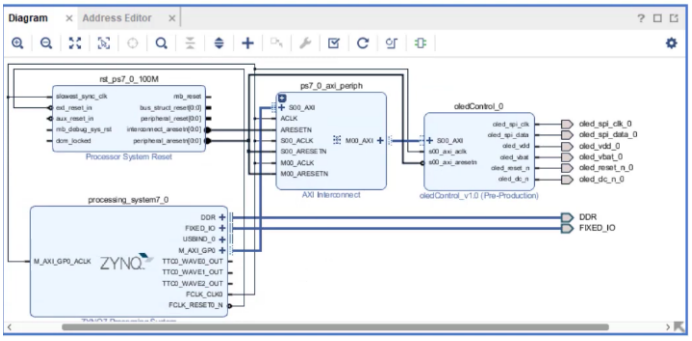

Fig. 4. block diagram

of the required steps.) The code, written in Verilog, includes states responsible for initializing the OLED. These states are invoked from the top.v module.

Initially, the display was tested locally, operating without the AXI bus, directly on the ZedBoard. The display successfully showcased ”Hello! World!” confirming the correct setup of the ZedBoard and OLED display. An observation during local testing revealed small dots scattered across the display, indicating that the RAM values were not reset properly.

The next step involved creating an IP block and connecting the ZedBoard to the AXI bus. To create the IP block, a new project was initiated to generate a block design. The oledControl was imported as a block into the design alongside the ZynqOLED system block. Xilinx automatically facilitated the connection of ports between the two blocks, resulting in a configuration resembling Figure 4. Subsequently, the oledControl_v1_0_S00_AXI.v file was edited to accommodate user-customized logic. A snippet of the edited code for user logic is presented in Figure 5.

Once the AXI bus connection was established, the SDK manager, the software driver on the opposite end of the AXI bus, was opened. On the software end, the project was coded in C, comprising three files: main.c, oled.c, and oled.h. The message to be displayed is stored in main.c, allowing for easy modification by altering the string within the main file. After updating the string in main.c, the driver communicated through the AXI bus to the OLED driver.

IV. Results and Discussion

The OLED was successfully displaying with ”nihao! Mitchellchi” after the author ran into a lot of problem when generating bit-stream and debugged it. The problem that the author ran into was when following Vipin’s video he skips a few steps, which led the authors not including the constraints file when building the ip block. The constriants should look like the following code for the ZYNQ and the OLED control ip to be successfuly generate bitstream onto the Zedboard. Without generating the bitstream file and programing onto the zedboard, the C program controller would not be able to control the zedboard which nothing would be transmitted through the AXI bus. This would defeat the purpose of the project and stopped the exploration between the automation programming and the local FPGA connection.

V. Conclusion

In conclusion, the project was successfully executed which not only deepened our understanding of the AXI bus and OLED display on the ZedBoard but also transformed theoretical research into practical work. Beyond these objectives, troubleshooting and debugging was also a huge lesson for this project, which highlights the real-world challenges in hardware design. Customizing logic showed me how powerful

```
    // Add user logic here

    top oledTop(
.clock(S_AXI_ACLK), //100MHz onboard clock
.reset(!S_AXI_ARESETN),
//oled interface
.oled_spi_clk(oled_spi_clk),
.oled_spi_data(oled_spi_data),
.oled_vdd(oled_vdd),
.oled_vbat(oled_vbat),
.oled_reset_n(oled_reset_n),
.oled_dc_n(oled_dc_n),

.sendData(slv_reg2),
.sendDataValid(slv_reg0[0]),
.sendDone(sendDone)
);
```

Fig. 5. snippet of oledControl_v1_0_S00_AXI.v file

FPGA are, while interaction between software and hardware components emphasized the nature of embedded systems. This report is a documentation of the things I did for this project and to show my own understanding after following the tutorial.

VI. Acknowledgment

R.C. would like to extend his gratitude to Professor Hoerning for his valuable guidance and supervision throughout the entire project, particularly in the areas of AXI bus and OLED display. Additionally, he would like to convey his appreciation to the ECE 311 Hardware Design for affording him the chance to engage with Xilinx Vivado and the ZedBoard. Lastly, he would like to show appreciation to Vipin Kizheppatt for his useful and insightful video documnetation on the OLED zedboard.

References

[1] https://support.xilinx.com/s/article/1053914?language=en_US
[2] https://www.youtube.com/watch?v=GILjTGScbfc

```
begin
    case(state)
        IDLE:begin
            oled_vbat <= 1'b1;
            oled_reset_n <= 1'b1;
            oled_dc_n <= 1'b0;
            oled_vdd <= 1'b0;
            state <= DELAY;
            nextState <= INIT;
        end
        DELAY:begin
            startDelay <= 1'b1;
            if(delayDone)
            begin
                state <= nextState;
                startDelay <= 1'b0;
            end
        end
        INIT:begin
            spiData <= 'hAE;
            spiLoadData <= 1'b1;
            if(spiDone)
            begin
                spiLoadData <= 1'b0;
                oled_reset_n <= 1'b0;
                state <= DELAY;
                nextState <= RESET;
            end
        end
        RESET:begin
             oled_reset_n <= 1'b1;
             state <= DELAY;
             nextState <= CHRG_PUMP;
        end
        CHRG_PUMP:begin
            spiData <= 'h8D;
            spiLoadData <= 1'b1;
            if(spiDone)
            begin
                spiLoadData <= 1'b0;
                state <= WAIT_SPI;
                nextState <= CHRG_PUMP1;
```

Fig. 6. oledControl code

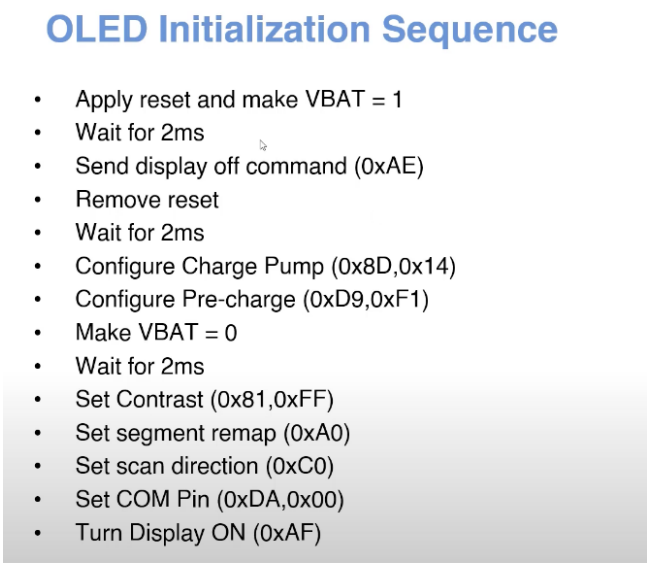

Fig. 7. OLED initialization

```
1   set_property PACKAGE_PIN U10 [get_ports oled_dc_n]
2   set_property PACKAGE_PIN U9 [get_ports oled_reset_n]
3   set_property PACKAGE_PIN AB12 [get_ports oled_spi_clk]
4   set_property PACKAGE_PIN AA12 [get_ports oled_spi_data]
5   set_property PACKAGE_PIN U11 [get_ports oled_vbat]
6   set_property PACKAGE_PIN U12 [get_ports oled_vdd]
7   set_property IOSTANDARD LVCMOS18 [get_ports oled_dc_n]
8   set_property IOSTANDARD LVCMOS18 [get_ports oled_reset_n]
9   set_property IOSTANDARD LVCMOS18 [get_ports oled_spi_clk]
10  set_property IOSTANDARD LVCMOS18 [get_ports oled_spi_data]
11  set_property IOSTANDARD LVCMOS18 [get_ports oled_vbat]
12  set_property IOSTANDARD LVCMOS18 [get_ports oled_vdd]
```

Fig. 8. Constraints file for the OLED display