

ECE385 Experiment #8

Eric Meyers, Ryan Helsdingen

Section ABG; TAs: Ben Delay, Shuo Liu

March 30th, 2016

emeyer7, helsdin2

I. INTRODUCTION

The purpose of this lab is to introduce concepts pertaining to USB protocol/communication and VGA display. The main goal of this lab was to connect a USB keyboard to the Altera FPGA and allow a user to control a ball displayed on a VGA-connected monitor.

II. DESCRIPTION OF CIRCUIT

RYAN SECTION

III. PURPOSE OF MODULES

RYAN SECTION

IV. DESCRIPTION OF USB PROTOCOL & CHANGES

The USB protocol in Experiment 8 utilized the Cypress EZ-OTG (CY7C67200) USB controller on board the Altera. The CY7C67200 was used as a host controller and once a USB keyboard is plugged in, the keyboard acts as a device controller.

The two functions that were written by the team were in "usb.c" and "io_handler.c" were pertaining to the reading and writing of data to and from the USB device controller (keyboard).

io_handler.c:

- void IO_init(void)
- void IO_write(alt_u8 Address, alt_u16 Data)

```
void IO_write(alt_u8 Address, alt_u16
Data)
{
    *otg_hpi_address = Address;
    *otg_hpi_cs = 0;
    *otg_hpi_w = 0;
    *otg_hpi_data = Data;
    *otg_hpi_w = 1;
    *otg_hpi_cs = 1;
}
```

```
alt_u16 IO_read(alt_u8 Address)
{
    alt_u16 temp;
    //printf("%x\n",temp);
    *otg_hpi_address = Address;
    *otg_hpi_cs = 0;
    *otg_hpi_r = 0;
    temp = *otg_hpi_data;
    *otg_hpi_r = 1;
    *otg_hpi_cs = 1;
    return temp;
}
```

usb.c:

- void UsbWrite(alt_u16 Address, alt_u16 Data)
- alt_u16 UsbRead(alt_u16 Address)

```
void UsbWrite(alt_u16 Address, alt_u16
Data)
{
    IO_write(HPI_ADDR, Address);
    IO_write(HPI_DATA, Data);
}
```

```
alt_u16 UsbRead(alt_u16 Address)
{
    IO_write(HPI_ADDR, Address);
    alt_u16 temp = IO_read(HPI_DATA);
    return temp;
}
```

V. SCHEMATIC/BLOCK DIAGRAM

ERIC SECTION

VI. POST LAB

1. What is the difference between VGA_clk and Clk?

Resource	Value
LUT	
DSP	
Memory (BRAM)	
Flip-Flop	
Frequency	MHz
Static Power	mW
Dynamic Power	mW
Total Power	mW

TABLE I: Design Statistics

of the section of code that bounces the ball so that there is no delay in processing.

VII. CONCLUSION

RYAN SECTION

Answer: The VGA Clock runs at 25MHz to change how wide the individual pixels are, whereas the Clk onboard the processor runs at 50MHz (and does not affect the pixel size).

2. In the file `io_handler.h`, why is it that the `otg_hpi_address` is defined as an integer pointer while the `otg_hpi_r` is defined as a char pointer?

Answer: `otg_hpi_r` is a single bit and does not need to be declared as an int (16 bits - it would be a waste of space), whereas `otg_hpi_address` is multiple bits wide and must have multiple bits available to use.

3. What are the advantages and/or disadvantages of using a USB interface over PS/2 interface to connect to the keyboard? List any two.

Answer: PS/2 keyboards aren't polled, but are completely interrupt based. This allows the processor to complete tasks while waiting. Drivers for PS/2 are much simpler than USB keyboard drivers. Another disadvantage is that the USB keyboard only takes in 6 keys every message, so this is limited compared to the PS/2. These are two disadvantages for using a USB keyboard over PS/2.

4. Note that `Ball_Y_Motion` in the above statement may have been changed at the same clock edge that is causing the assignment of `Ball_Y_pos`. Will the new value of `Ball_Y_Motion` be used, or the old? How will this impact behavior of the ball during a bounce, and how might that interact with a response to a keypress? Can you fix it?

Answer: The new value of `Ball_Y_Motion` that would be used would be the old one and this would make it so that the ball bounces one clock cycle after the keypress is handled. A fix would be to put the assignment inside

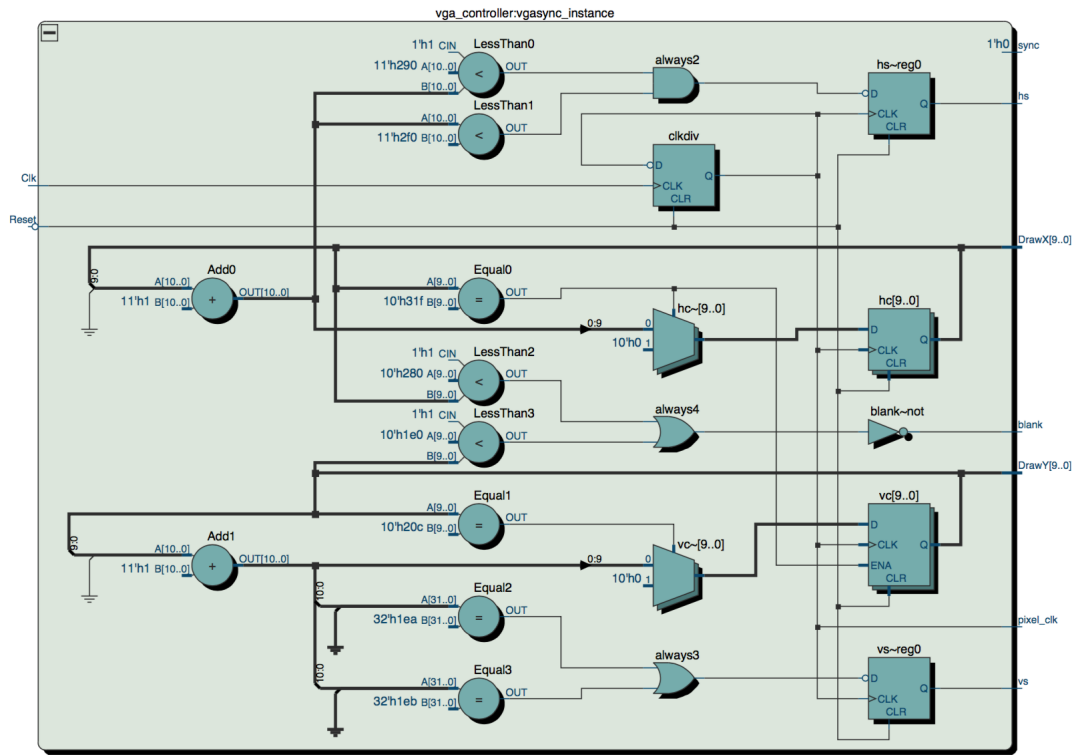


Fig. 3: VGA Circuit Diagram

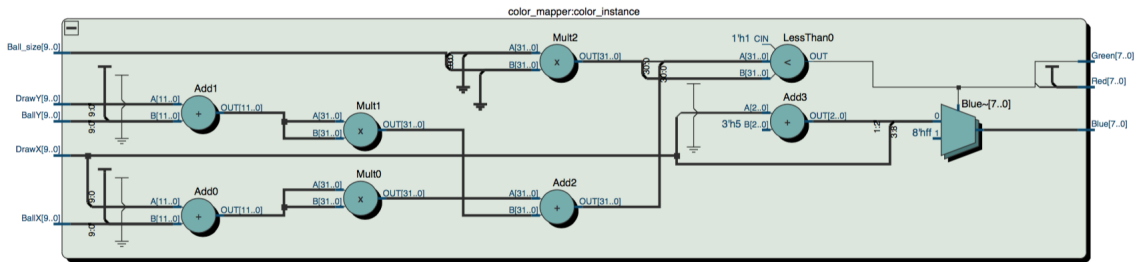


Fig. 4: Color Mapper Circuit Diagram

APPENDIX