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 USB Keyboard is not an interrupt-based device, rather the host must poll the keyboard and send requests to receive the scancode in return. For this, the team must write functions to perform input/output reading and writing. These functions must be used in conjunction with the system verilog hardware that is synthesized to retrieve data and write data.

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;
}
```

This function takes in parameters as the Address and Data to be written. It then sets the appropriate bits on the signal lines and sends the Data to the otg_hpi_data variable to be written to the usb hardware.

```
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;
}
```

This function performs similar actions to the IO_write function in that it must read from a particular address on the USB hardware. The team had to use a temp variable to read it properly, set the appropriate signals, then send the data from otg_hpi_data into the temp variable which is then returned to the user by the function.

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);
}
```

This function utilizes the two functions written in the previous section (in io_handler.c) and simply writes the particular address and the data in the parameters.

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

This function uses the two functions written in io_handler.c to perform a read on the USB hardware. it first must write the address to HPI_ADDR to receive the proper address, then the temp variable must store the result of IO_read from HPI_Data.

V. Schematic/Block Diagram

The Schematic / Block Diagrams for this lab can be found in the Figures section of this document ("Section XI: Figures"). The Top Level Module Diagram can be found in Figure 1, the Ball Module Diagram can be found in Figure 2, the VGA Module Diagram can be found in Figure 3, and the Color Mapper Module can be found in Figure 4.

VI. Post Lab

Resource	Value
LUT	2663
DSP	10
Memory (BRAM)	82,944
Flip-Flop	646
Frequency	132.89 MHz
Static Power	102.09 mW
Dynamic Power	25.86 mW
Total Power	203.64 mW

TABLE I: Design Statistics

1. What is the difference between VGA_clk and Clk?

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

VII. Conclusion

RYAN SECTION

VIII. Figures

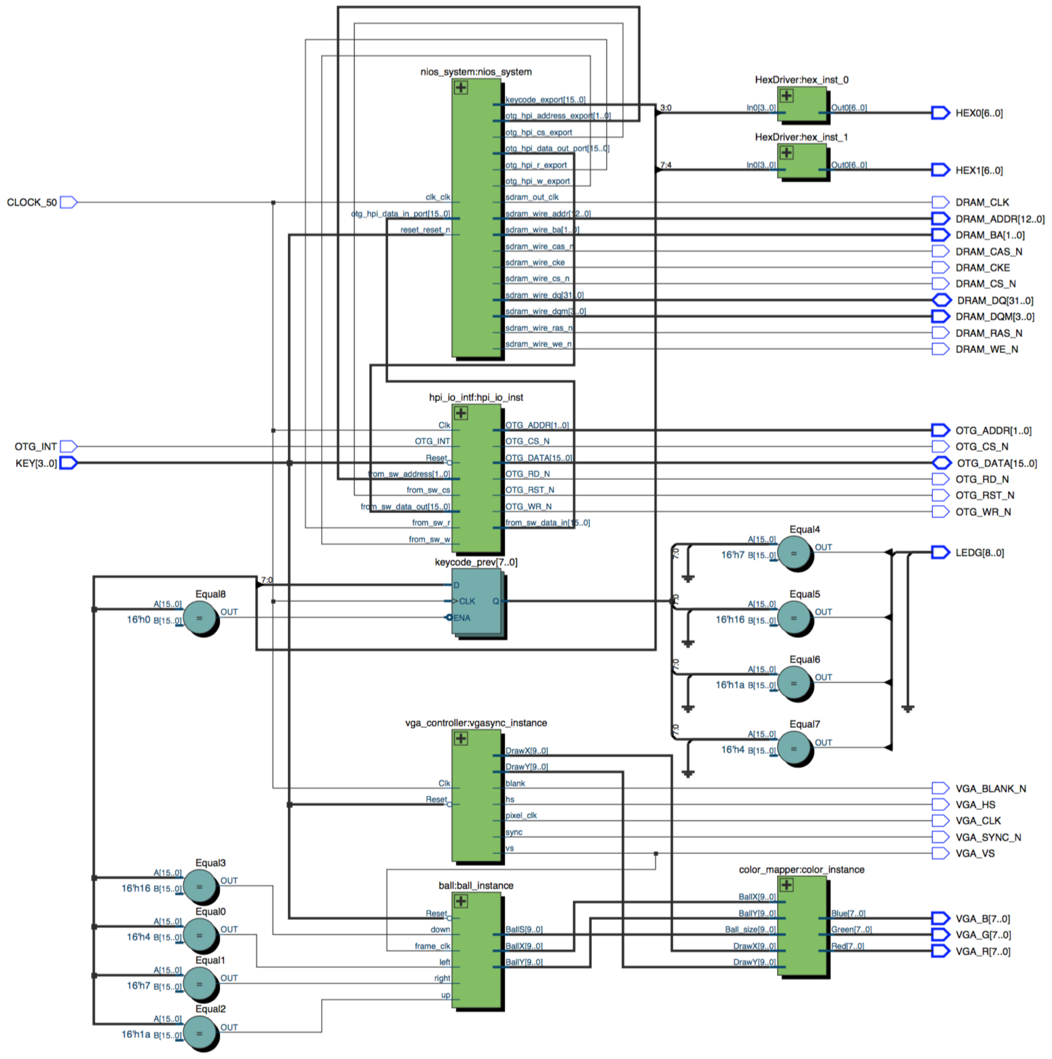

Fig. 1: Lab 8 Top Level SV

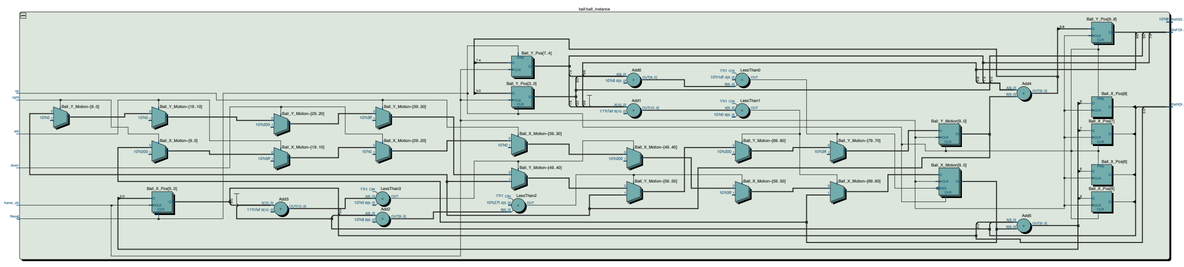

Fig. 2: Ball Circuit Diagram

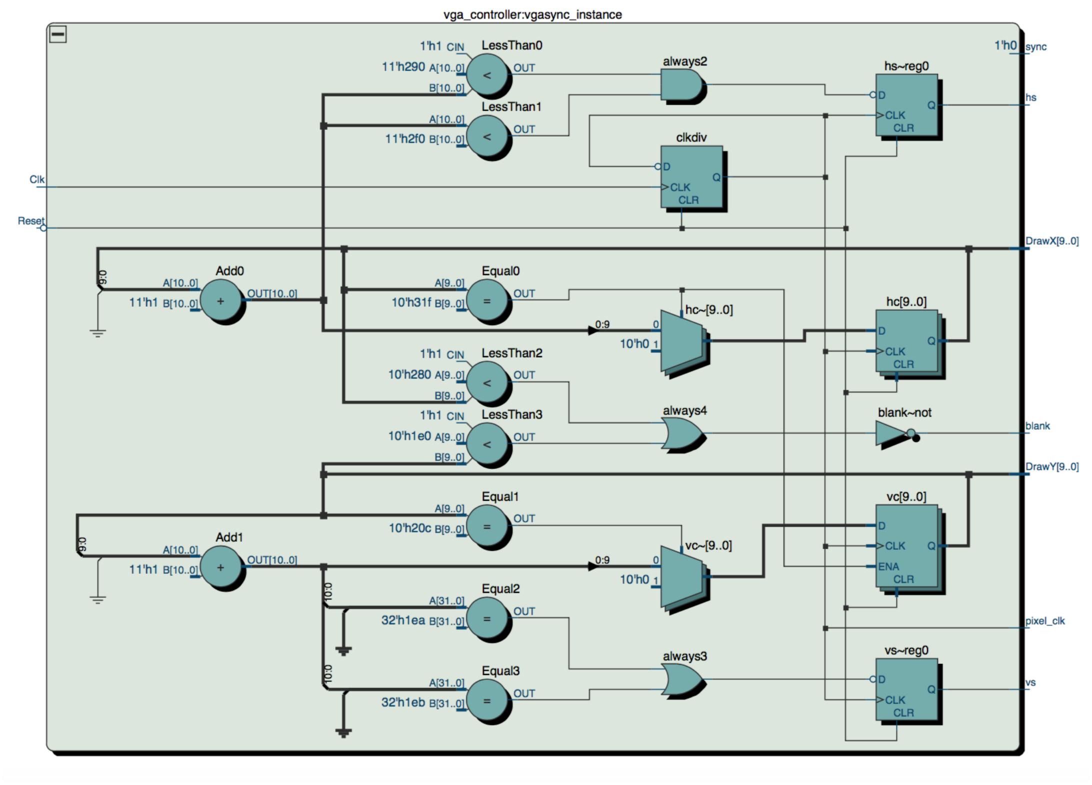

Fig. 3: VGA Circuit Diagram

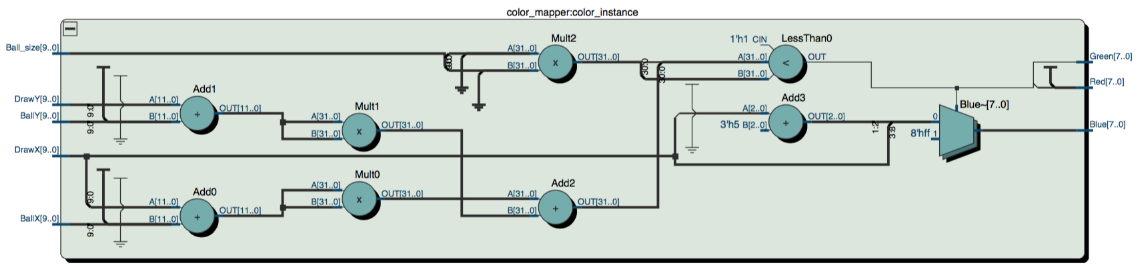

Fig. 4: Color Mapper Circuit Diagram

APPENDIX