USB Controller and Drivers

Submitted by Group 4 (Multi-touch table): Alix Krahn, Denis Lachance, Adam Thomson

The generic USB controller and driver did not function when we tested it. We created a new hardware controller, as well as part of a software driver, for USB. The software driver was not completed, due to time constraints, but we wanted to pass along the code for the controller and driver. Currently, the software driver can sense that a device is connected, and can send and receive control packets. Some code is included to receive isochronous data packets, but this code was never completed.

The USB is controlled through the ISP1362 chip, a single-chip universal serial bus on-the-go controller. The ISP 1362 has two USB ports. Port 1 can be hardware configured to function as a downstream port, an upstream port, or an OTG port. Port 2 can only be used as a downstream port. The OTG port can switch roles from host to peripheral, or from peripheral to host. The host and device controllers are fully compliant with *USB 2.0 Specification Rev. 2.0.* The ISP1362 supports data transfer at full-speed (12Mbit/s, or 1.5MB/s) and low-speed (1.5Mbit/s, or 187.5kB/s). Note that this data transfer speed is not the high-speed standard (480Mbit/s, or 60MB/s), and may be too slow for high quality images or video.

In our use, the Altera DE2 is used as a host to a webcam, using port 1, using isochronous transfers.

Archived Project

1. The project is archived. Open the file, and, in Quartus, restore the project. It includes the Qsys system and VHDL files (component and top-level). The Qsys is appropriate for both volatile and non-volatile systems. Note that three warnings to do with the altpll are normal in the Qsys.

Qsys System

To add in the component to your existing system, follow the below steps.

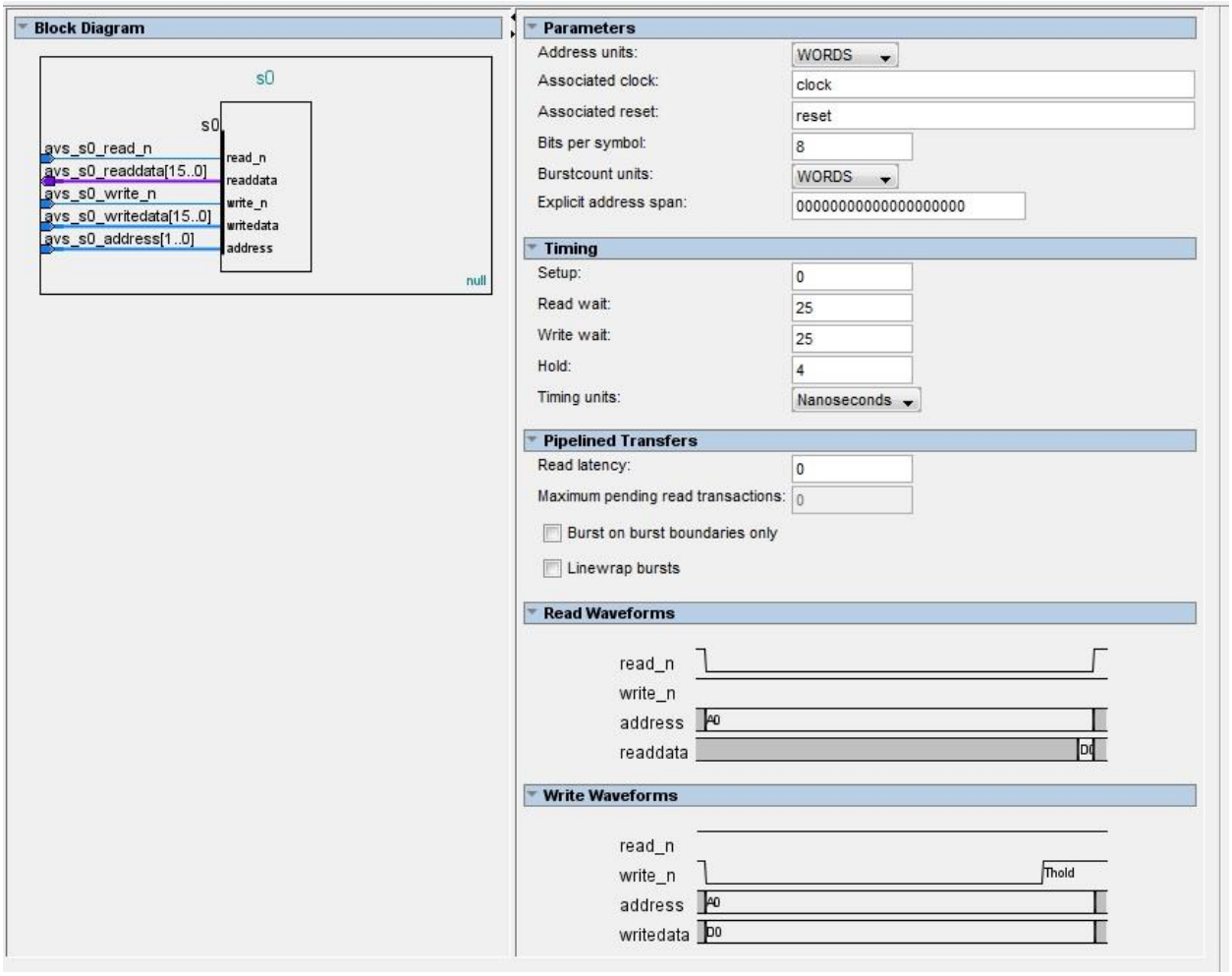
1. Use an existing design, or create a new system and add in the usual components (NIOSII processor, JTAG UART, On-Chip Memory, System ID, and SRAM controller).

2. Add in the VHDL component (usb_controller.vhd), in addition to the DE2_constants.vhd file to the project.

3. In Qsys, add a new component. In the Files tab of the Component Editor, include the two vhd files (usb_controller and DE2_constants), and then click “Analyze Synthesis Files”.

4. In the Signals tab, ensure that there are five interfaces: clock, reset, s0, conduit_end, and interrupt_sender. The conduit_end interface should all of have type of expor. For other interfaces, the signal type is the same as the name. The interrupt is irq_n.

5. In the Interfaces tab, under “s0”, ensure that the parameters and timing are as below.



6. Make the connections required to the rest of the system (clock, rest, s0 should be connected to the data master of the NIOSII).

7. Double click on the out conduit line of the USB controller component under the Export column to export the connections to the USB.

8. Under the IRQ column, click on the interrupt from the USB.

9. Generate the Qsys system.

Top-Level File:

1. Modify your top-level *.vhd file to include the new USB ports. A sample top-level file is also attached. Include the following ports:

```
OTG_ADDR : out std_logic_vector(1 downto 0)
OTG_CS_N : inout std_logic_vector(0 downto 0)
OTG_RD_N : out std_logic_vector(0 downto 0)
OTG_WR_N : out std_logic_vector(0 downto 0)
```

```
OTG_RST_N  :    out std_logic_vector(0 downto 0)
OTG_DATA   :    inout std_logic_vector(15 downto 0)
OTG_INT0   :    in std_logic_vector(0 downto 0)
OTG_INT1   :    in std_logic_vector(0 downto 0)
OTG_DACK0_N :   out std_logic_vector(0 downto 0)
OTG_DACK1_N :   out std_logic_vector(0 downto 0)
OTG_DREQ0  :    in std_logic_vector(0 downto 0)
OTG_DREQ1  :    in std_logic_vector(0 downto 0)
OTG_FSPEED :    out std_logic_vector(0 downto 0)
OTG_LSPEED :    out std_logic_vector(0 downto 0)
```

Tie OTG_DACK0_N, OTG_DACK1_N, OTG_FPSEED, and OTG_LSPEED to "Z".

2. Compile your project after you modify the file.

Software Drivers:

1. Import the software files. The file hello.c sets up the device, USB.c and USB.h contain functions for setting up and accessing the USB device. The file qc_driver.c and qc_driver.h are not immediately applicable to any USB project (they are for the QuickCam Express Webcam), but may be useful to see how to write a USB driver.

2. After regenerating the bsp, rebuild the project.

3. Useful functions that were written by group 4 in 2014 include (all of these are found in USB.c/USB.h):

```
setup_controller()
wait_for_device()
print_device_info()
get_string_descriptor()
send_control_transfer()
get_control_transfer()
set_interface()
print_interface_info()
get_iso_transfer()
set_config()
set_address()
```

4. To write a driver for any USB device, use the code (found in main() in hello.c):

```
usb_device dev;
if (setup_controller() < 0) {
      printf("Setup Error!\n");
      return 0;
```

```
}
wait_for_device(&dev);
print_device_info(&dev);
```

This code will set-up a struct for the USB device, set-up the ISP1362 controller, and then wait for a device to be plugged in (and then print the device information).

5. Once that is completed, you can start accessing the USB device by sending commands, including:

```
set_config(usb_device *handle, int val);          // To handle a
configuration request
set_interface(usb_device *dev, int interface, int alt);      // To
set the interface
send_control_transfer(usb_device *dev, int bmReqType, int bReq,
        int wValue, int wIndex, unsigned char *data, int dataLen);
get_control_transfer(usb_device *dev, int bmReqType, int bReq,
        int wValue, int wIndex, unsigned char *data, int dataLen);
```

6. There are four classifications of functions in USB.h:
    a. Porting functions, used by old functions
    b. Old functions from ISP1362 documentation
    c. Modified functions to work with new functions written by group 4
    d. New functions written by group 4

7. The usb_device struct is new, written by group 4, to encapsulate all of the information for a USB device.

Recommendations

1. Use wireshark to trace the USB transfers. It uses the same types, and will tell you the lengths and such to use with the functions above.
2. Study the USB specification [2].

Source Documentation and References

[1] Philips, “Single-chip Universal Serial Bus On-The-Go controller,” ISP1362 datasheet, Dec. 2004.

[2] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips, *Universal Serial Bus Specification (2.0)* (Apr. 27, 2000). Available online: http://www.usb.org/developers/docs/usb20_docs/