ECEN3763 Lab Weeks 9, 10, 11
Full Resolution Frame Buffer using SDRAM
Spring, 2022
Due April 8th
105 points

Summary:

You will take all the information and experience from previous Labs and Homework assignments to construct a frame buffer system using the SDRAM on the DE10-Standard board. A Tcl script will run in System Console to initialize and start the video transfers. SignalTap will be used to capture video traffic read to or written by the DMA controller.

Submission and Grading:

Create a Quartus archive file (.qar) that contains your entire design, including Tcl script. When you run Project > Archive project in Quartus, the Archive Project window opens. If you press the Advanced button, you can see the complete list of files included in the archive. If you don't see your Tcl script in the file list, add it to the archive. Please do me a favor – add the .cdf file created by the Quartus programmer. This will save me a lot of mouse clicks as I grade this lab.

Please include a 1 to 2 page informal report discussing the problems you ran into, areas of the project that confused you, and if you were not able to complete the lab, the status of the work submitted.

This lab will be graded as follows:

Quartus project that compiles without error, containing Platform Designer frame buffer and your video controller	50 points
Tcl script to initialize and start video transfer	35 points
Signal Tap waveform capture showing Avalon transactions from the DMA controller	20 points

Description:

Create a full 720p60 24 bits/pixel resolution frame buffer for your video controller project. Use the JTAG Master Bridge to write some color or pattern into the SDRAM for display, program the DMA controller registers, and start the video traffic.

Suggestions:

1. Your Platform Designer component should include a PLL, JTAG Master Bridge, DMA Controller, SDRAM Controller, and a FIFO. Details on the DMA Controller and FIFO are provided later in this document.

2. Begin by constructing the frame buffer hardware in Platform Designer. Connect the Platform Designer component to you video controller so you have a complete project to compile and test.

3. Verify correct operation of SDRAM, and that you can communicate with the DMA controller registers. Since the DMA is reading data in 32 bit chunks, but the SDRAM is only 16 bits wide, make sure you understand how the data is ordered coming out of the SDRAM.

4. Use SignalTap to monitor Avalon interface signals as needed. You can construct multiple SignalTap cores, and enable and disable the cores as needed. You do this in Quartus using Assignments > Settings, select the Signal Tap Logic Analyzer category on the left side of the screen, and then select which core or cores to include (or exclude).

5. Pay attention to messages in the Quartus Transcript window, these are useful for finding design errors.

6. If you add the .qip file from your Platform Designer component as a source in Quartus, Quartus will not need to recompile the component each time. If you instead add the .qsys file, Quartus will recompile the Platform Designer component each time, adding to time waiting on Quartus.

Creating the FIFO

You want to use the Avalon FIFO Memory Intel FPGA IP component. Settings for the FIFO are:

Dual clock mode

Input type is AvalonmMM_write
Output type is AvalonST_source
Bits per symbol is 32
Symbols per beat is 1
Error and channel width can be 0
Enable packet data should be unchecked

Note: I would set the depth such that the FIFO can hold at least one full line of pixels

Creating and Programming the DMA Controller

You want to use the DMA Controller Intel FPGA IP component. Settings for the DMA controller are:

The Transfer size register must be big enough to hold the value of the number of bytes per frame.

Note: Do not enable burst transfers.

The FIFO depth can be left at the default value, although a larger value will probably improve performance.

Use word (32 bit transfers)

Using System Console to load video data into the SDRAM and to program the DMA controller registers is not difficult. However, determining the DMA controller register settings is challenging. You will find the Embedded Peripherals IP User Guide on Canvas (ug_embedded_ip.pdf is in the Files > Reference Material > Quartus folder). Chapter 29 provides information on the DMA controller.

The DMA Controller register map is shown in Table 282 on page 321. You will need to program the readaddress, writeaddress, length, and control registers. Reading the status register may help provide information to assist with debugging.

readaddress register: Program the register with the first address of the SDRAM where you wrote the pixel data, probably address 0.

writeaddress register: Program the register with the address of the FIFO you want to write to, again probably address 0.

length register: How many bytes will you read?

control register:

Bit 0: do not specify byte transfers

Bit 1: do not specify halfword transfers
Bit 2: select this for word transfers
Bit 3: This bit is to be set in a separate step after all DMA registers are programmed.
Bit 4: If you want to monitor the done bit with an LED, you could enable interrupts and tie the DMA irq controller signal to an LED on the board.
Bits 5 & 6: Set to 0
Bit 7: Set to 1
Bit 8: Set to 0 so SDRAM address increments during reads
Bit 9: Set to 1 so DMA writes all go to a single address (FIFO)
Bits 10 & 11: Set to 0
Bit 12: Set to 0