Maxime Grégoire

260479270

Patrick White

260353491

Final report

ECSE 431

Introduction to VLSI CAD

McGill University

December 10th 2013

**Description of the design**

The FPGA is used to display images coming from a standard NTSC signal through the video decoder on a graphic monitor. The image is temporarily stored into memory before being displayed. The grab interface writes the image into memory, the NIOS processes it, and the DMA engine reads it and feeds the VGA controller that displays it. The NIOS also executes the driver code that sequences the acquisition *field by field*. The Qsys system operates at 100MHz.

**DMA engine design**

The DMA engine we implemented acted as an Avalon Master controller that writes data into a line buffer, as mentioned in the specifications. The engine was designed after the given block diagram.

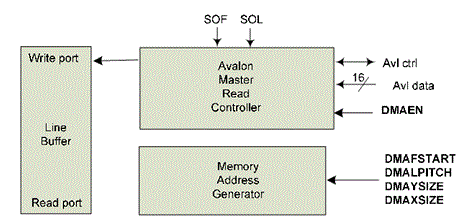
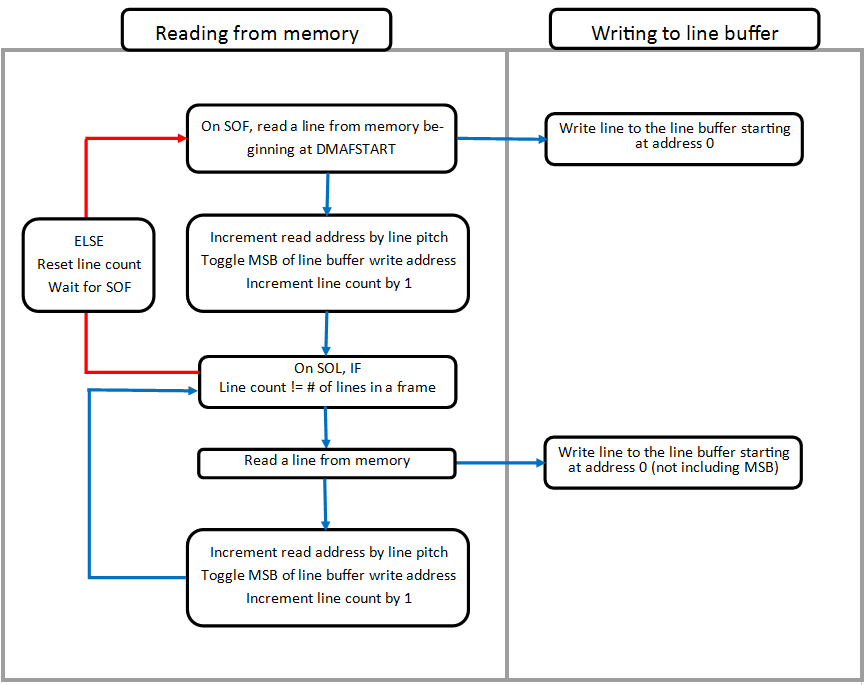


Figure 1 - DMA Engine Block Diagram

The DMA engine sends bursts of 120 bytes to the line buffer upon receiving the SOF and SOL signals (see appendix, Figure 4, Figure 5, Figure 6). It can write a full frame into the line buffer (see appendix, Figure 7).



Note that the code for the "dma\_engine.vhd" code is in the "code.zip" archive, attached to this report.

**Modification to VGA module**

The VGA controller used in the design was a modified version of one that was provided to us. The original VGA controller was a slave in its original implementation. In this implementation, the VGA controller is essentially the master of the DMA engine. The modification required to make this possible is for the VGA controller to generate SOF and SOL single clock cycle pulses to indicate to the DMA engine when to fetch and write lines to the line buffer. The SOF signal is generated on a falling edge of the VSyncN signal and indicates that a new frame has begun. The SOL signal is generated on a falling edge of the HSyncN signal and indicates that a new line has begun within the frame. The SOL signal is also used to toggle the upper most bit of the read address sent to the line buffer.

In simulation, the ADV7181b decoder provided a pixel ramp. The output of the VGA controller for this pixel ramp can be seen in Figure 12. The RGB values correspond to the ramp shifted by left by 2 bits.

**C program and initialization**

For an easy switch between simulation and hardware modes, two set of values have been defined for the initialization of the register file. To change the mode, we simply change the value of the "sim" variable to either "SOFTWARE" or "HARDWARE".

Here is a list of the actions done by the NIOS II:

1. It writes the fields of the register file (see appendix, Figure 8)
2. It registers the IRQ (interrupt request)
3. It enables the DMA engine and writes its parameters
4. It sets a snapshot (GSSHT register)

The Interrupt Service Routine occurs when there is an "End Of Frame" (see appendix, Figure 9). It does the following things:

1. Resets the SOFISTS and EOFISTS signals
2. Toggle the GMODE from even to odd frame or odd to even frame
3. Updates GFSTART and DMAFSTART registers to a new line location (see appendix,
4. Sets the GSSHT signal (see appendix, Figure 10)

The complete code can be found under the name "FPGA.c" in the "code.zip" archive attached to this report.

**QSYS component**

The address map of the QSYS components is defined as follows:

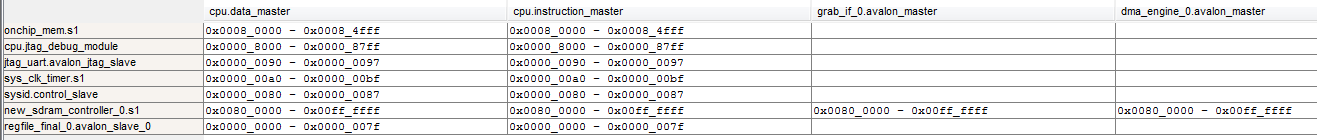


Figure 2 - Address map of the QSYS components

The connections of the QSYS components are defined below.

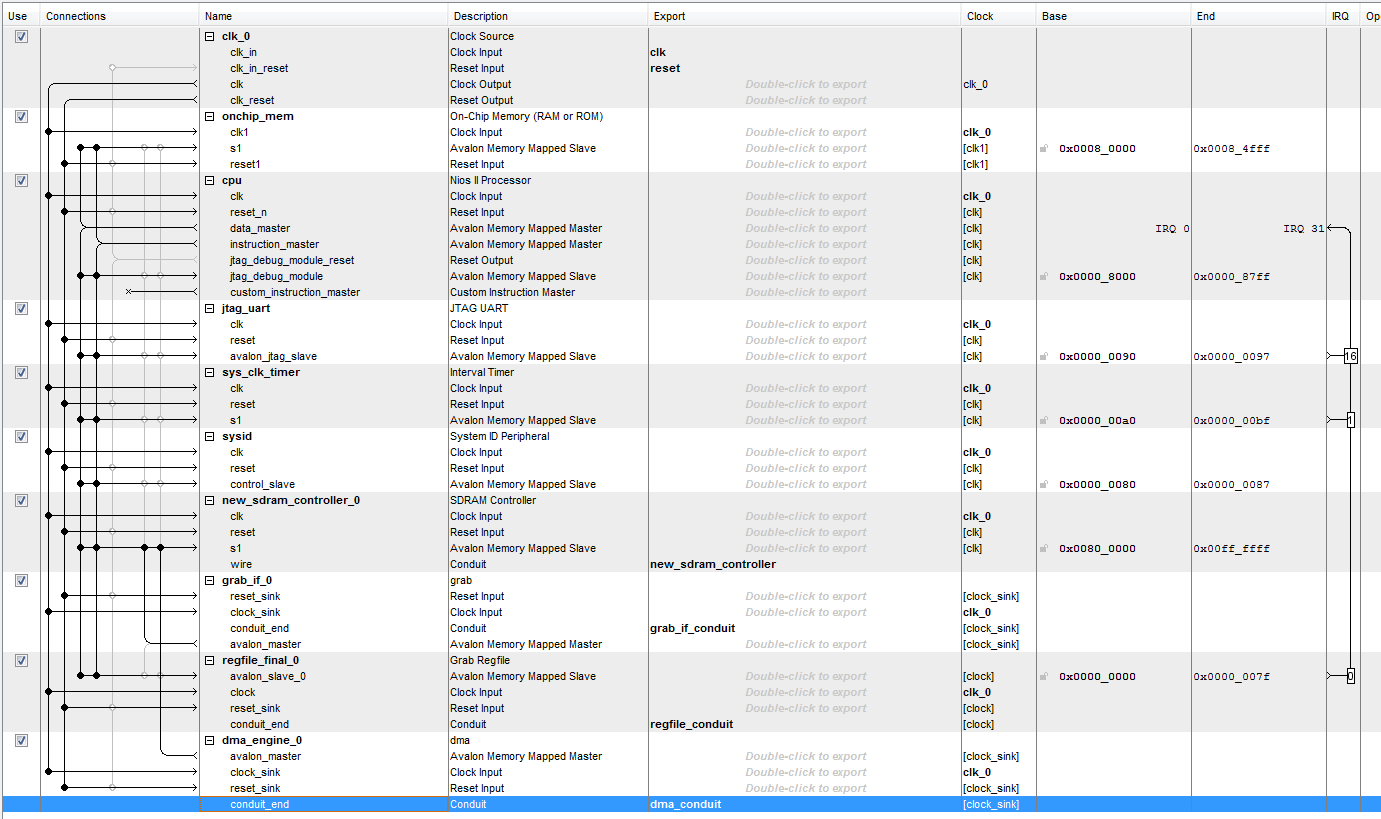
****

Figure 3 - The connections of the QSYS components

**Processing**

Since we were only two students to do this project, we could not get any image processing done in time.

**Constraint file**

The full constraint file used to compile our design can be found under the name "Real constraint file.txt" in the "code.zip" archive attached to this report.

**Compilation reports**

The screenshots of the compilation reports can be found in Figure 11 through Figure 19 of the Appendix.

**Sources of problems**

TODO

The grab interface not writing enough

**APPENDIX**

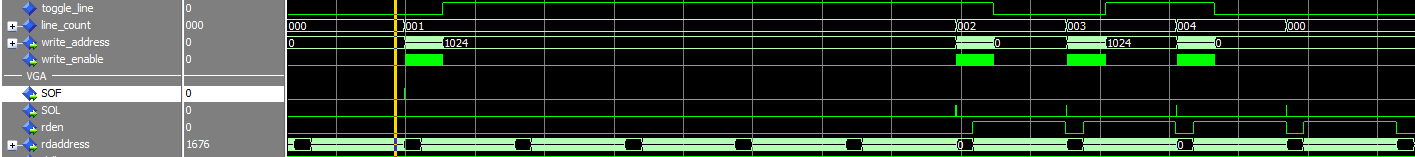


Figure 4 - When receiving SOF and SOL, the DMA Engine writes into the line buffer,

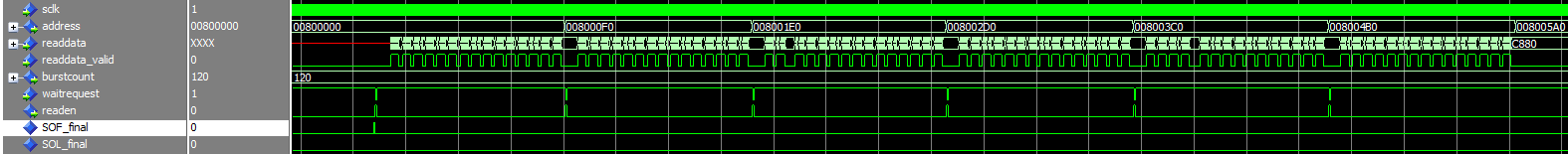


Figure 5 - When receiving SOF signal, the DMA engine writes into the line buffer

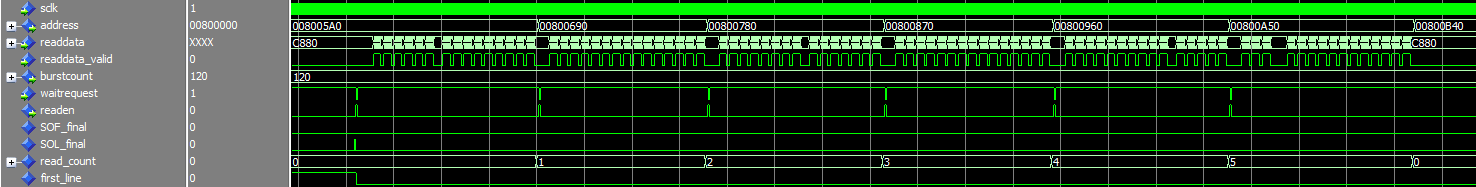


Figure 6 - When receiving SOL, the DMA engine writes into the line buffer

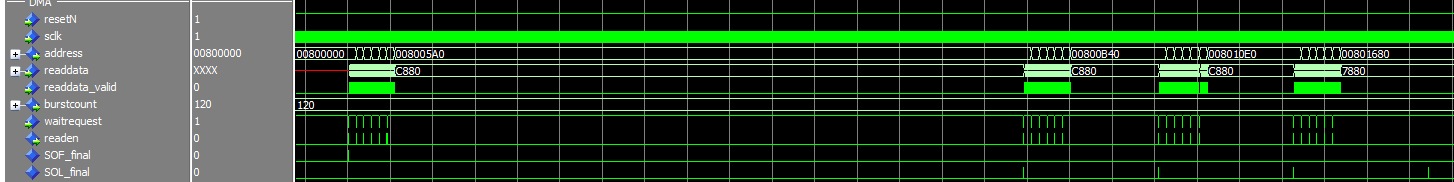


Figure 7 - The DMA engine writes a full frame of 4 lines

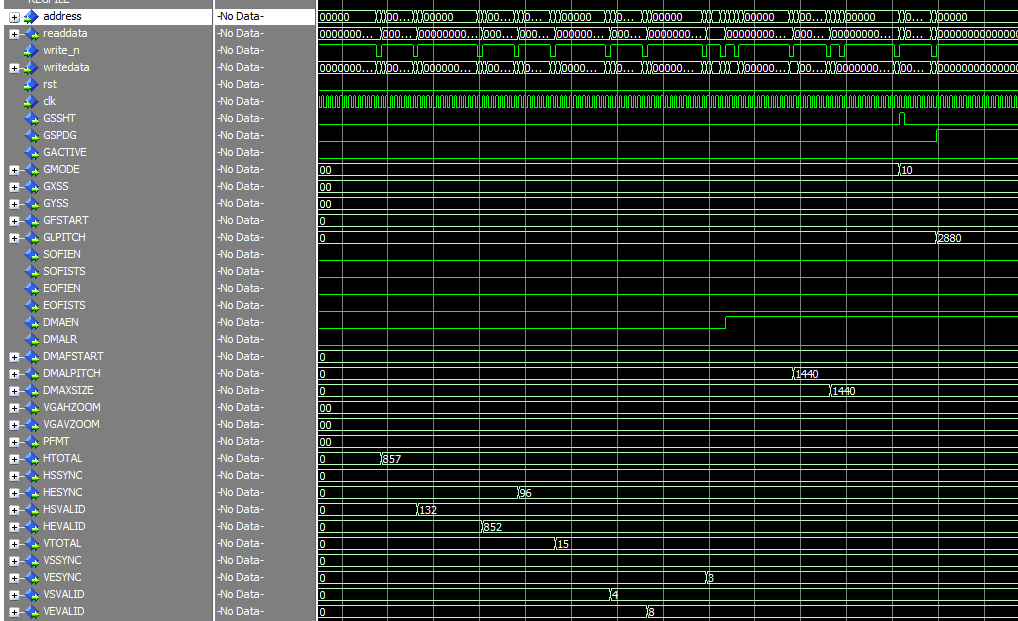


Figure 8 - Initialization of the register file by the NIOS II

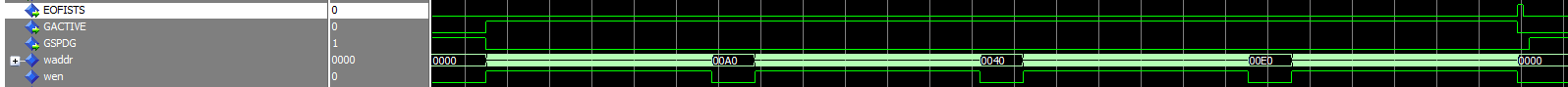


Figure 9 - End Of Frame Interrupt

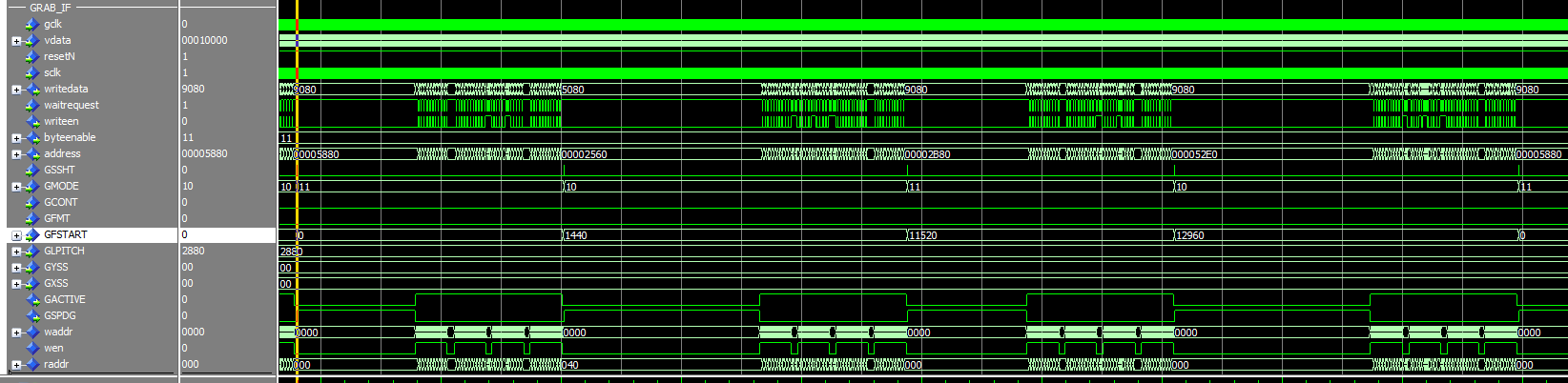


Figure 10: 2 Full Frames in the Grab Interface

C:\Users\pwhite8\vlsi\Screenshot\Setting a snapshot.png

Figure 11 - Setting a snapshot

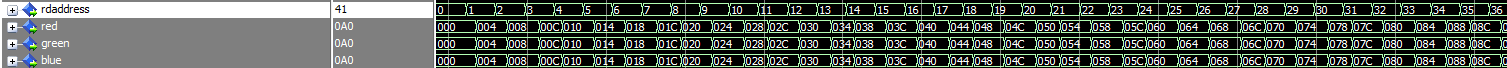


Figure 12: Pixel Ramp in VGA Controller

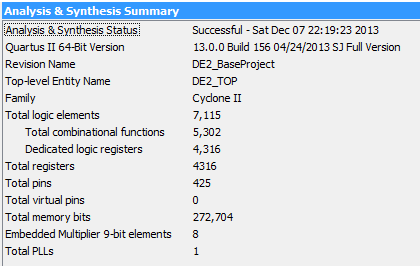


Figure 13 - Analysis & Synthesis Summary

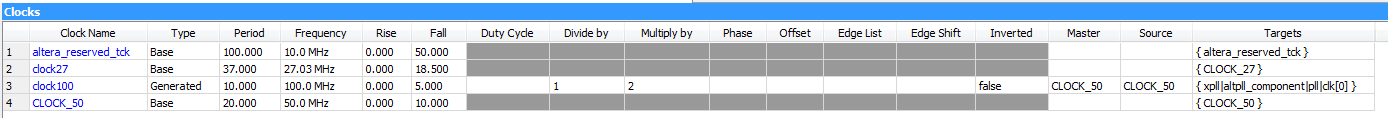


Figure 14 - Clocks present in our design

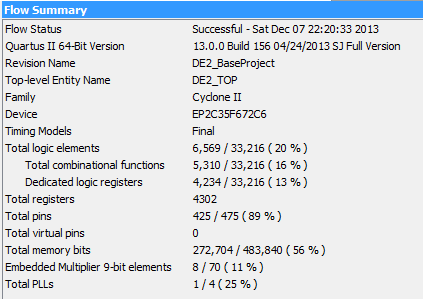


Figure 15 - Flow Summary

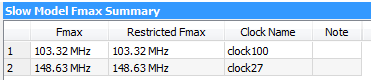


Figure 16 - Slow Model Maximum Frequency

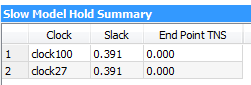


Figure 17 - Slow Model Hold Summary

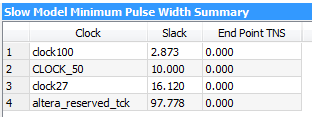


Figure 18 - Slow Model Minimum Pulse Width

C:\Users\Max\vlsi\Screenshot\Compilation Report\Slow Model Recovery.png

Figure 19 - Slow Model Recovery Summary

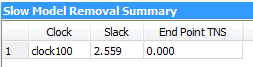


Figure 20 - Slow Model Removal Summary

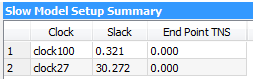


Figure 21 - Slow Model Setup Summary

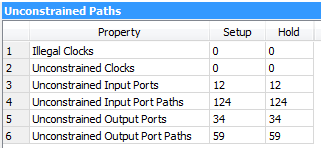


Figure 22 - Unconstrained Paths