ECEN2350 Digital Logic – Project 4

Fall, 2020

Due Monday, December7th, 2020, end of day

75 points

Project4 has been modified in order to reduce the amount of time required to complete the project, so that everyone will have time to complete this. Each student must turn in their own submission. There will be no weekly submissions required for Project 4.

The goal of Project4 is to analyze a video controller design using simulation, and using the SignalTap logic analyzer.

Unlike the previous 3 projects, you do not have to write a project report. You must answer all the questions in this document, and also add screenshots as requested. Use this document to answer your questions. When you are finished, convert your final document to a .pdf file and submit the document to Canvas. Please include your name in the document title.

You will be required to answer a number of questions as you work through this project, as well as create a testbench and run simulations. Finally, you will modify the project by including an embedded logic analyzer called Signal Tap into the design. As you work through the project, you will see items in red. **The red color signifies that you must either answer questions or include screen captures for submission.**

Part 1: Understanding VGA and the Video Controller Circuit Operation

VGA or Video Graphics Array is an analog video standard used originally by IBM personal computers. The standard was introduce in 1987, and is still in use today. You DE10-Lite board has a 15pin D-type connector that is the VGA standard connector, and the factory program that runs on your board when first plugged in outputs a 640 pixel x 480 line VGA signal. If you have an older monitor that can connect to a VGA cable (and the cable) you could test this out. Or, you can purchase a VGA to HDMI or DVI adapter that would allow the DE10-Lite to drive a signal to your monitor.

The VGA standard closely mimicked NTSC (old, analog television) television timing. NTSC dictated that a television picture consisted of 525 total lines of video, at a typical resolution equivalent 640 pixels per line. One major difference is that NTSC video would first draw the odd numbered lines, and then draw the even numbered lines. This approach was called interlacing, and resulted in a complete picture being drawn at 30 frames / second. The persistence of the TV cathode ray tubes allowed this approach to produce a picture quality that

was not a problem for the human eye to view. However, computer graphics requires a higher quality image, so interlacing is not done.

The diagram below shows the circuitry on the DE10-Lite board that is used to generate a VGA video signal. The FPGA generates a horizontal synch(ronization) signal that is used for timing of each line of pixels. A vertical synch signal is used to signal the start of a new frame (or screen) or data. The red, green and blue signals are each 4 bits wide. A resistor network is used to convert the 4 bit red, green, and blue (RGB) digital signals to analog levels that drive the VGA connector. All timing for a VGA interface is contained in the horizontal and vertical synch signals. By adjusting the values of the RGB signals, you can produce 2^4 x 2^4 x 2^4 or 4096 different colors.

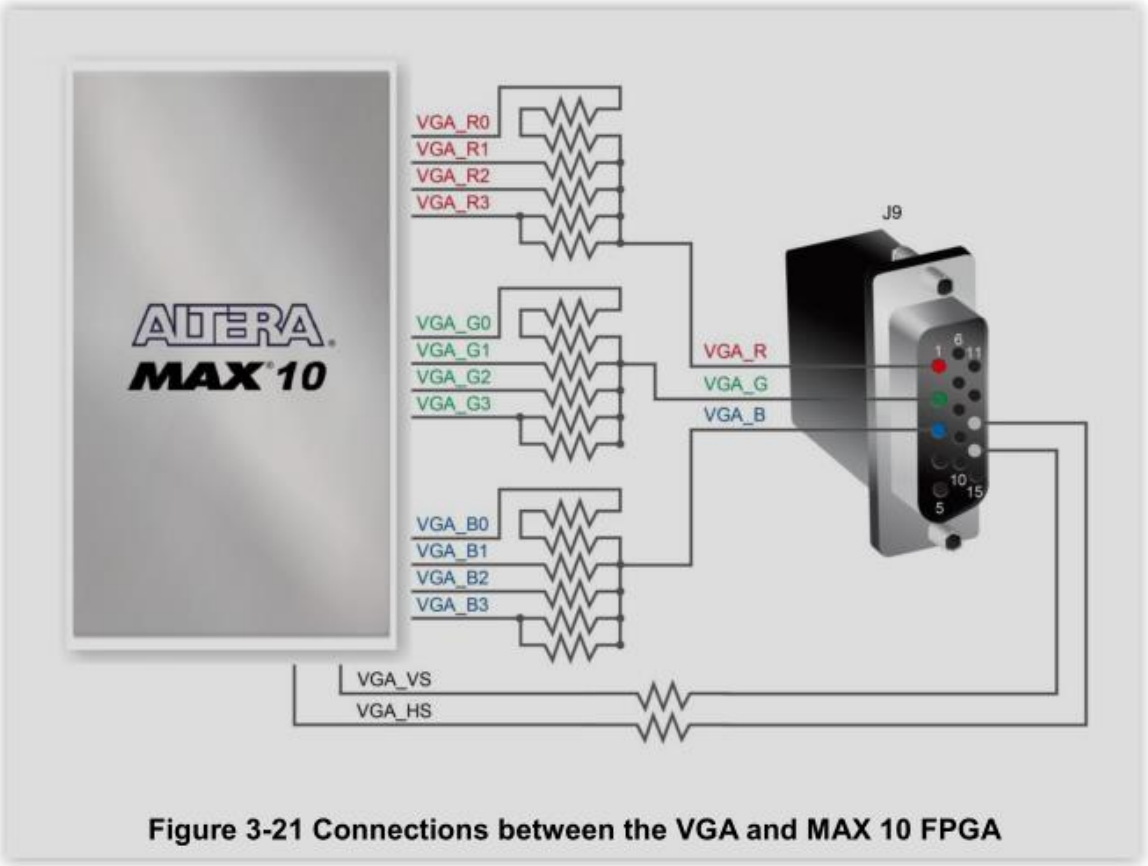
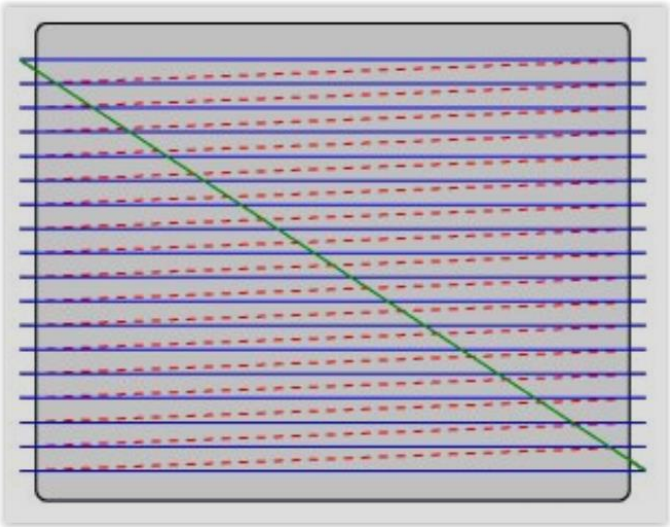


Figure 3-21 Connections between the VGA and MAX 10 FPGA

VGA resolution is defined as the number of visible pixels displayed in a line and the number of visible lines. 640 x 480 VGA drives 640 pixels and 480 lines of visible video. At the beginning of a frame, 640 pixels are painted horizontally left to right. At the end of a line, the line count increases by one, and a second line is displayed. This goes on until the last line is displayed, at which time the vertical synch instructs the monitor to reset to the top of the frame. The number of times per second a complete frame is redrawn is called the refresh rate.

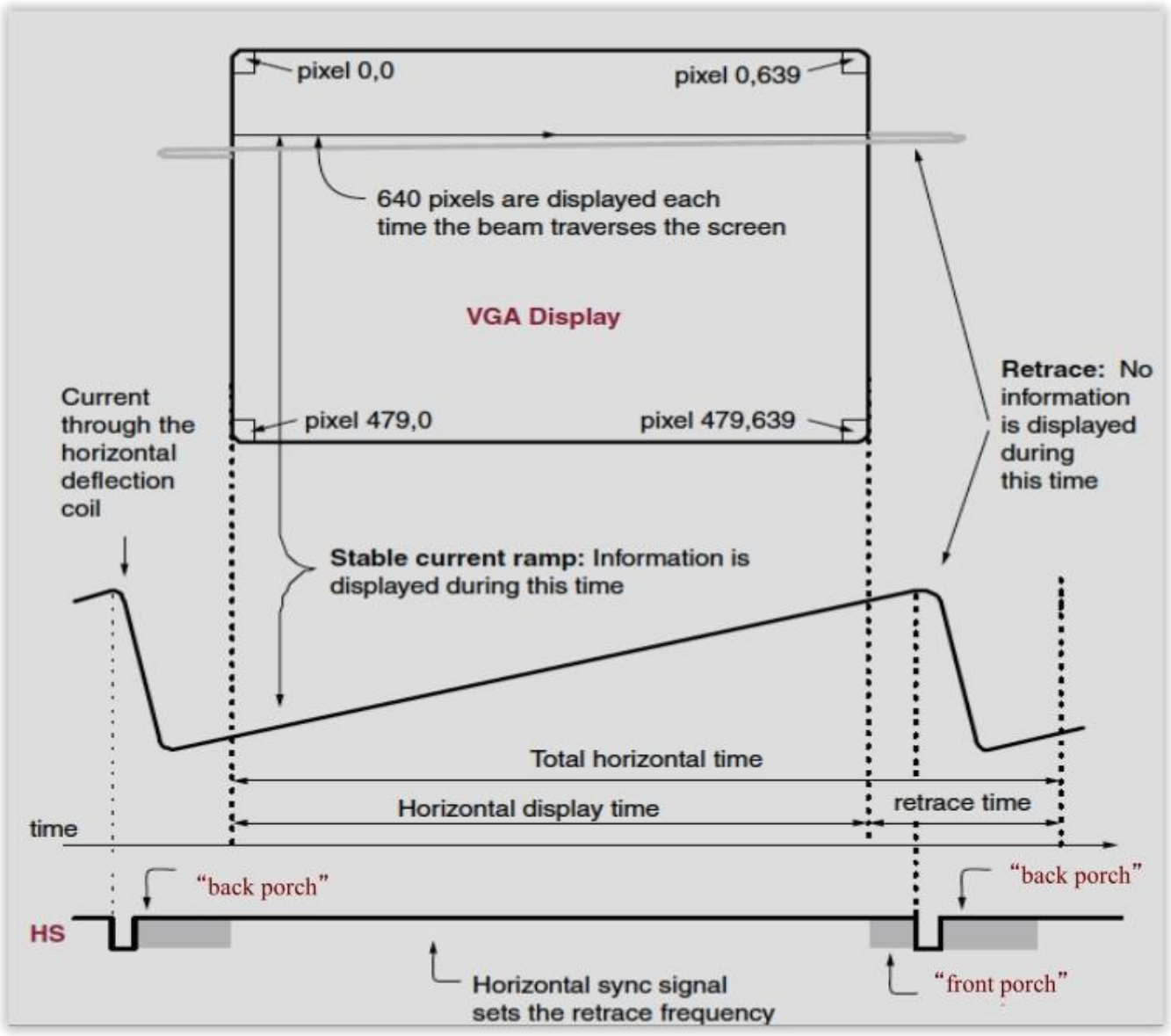
Below is an image of a non-interlaced display.



Since VGA was intended to drive older cathode ray displays, timing was needed to manage the retracing of the drive signal for both lines and frames. This requirement means that a single line of VGA contains more pixel times than visible pixels. This is also true for the number of lines. 640 x 480 VGA actually contains 800 pixel times per line and a total of 525 lines. These extra pixel times and lines are used to generate the horizontal and vertical synch signals, and active video is not output during these times.

The image below shows the timing required for a 640 x 480 VGA signal. After each horizontal line is displayed, the video controller waits for a period of time referred to as the horizontal front porch. After waiting long enough, an active low horizontal synch signal is generated for 96 pixel times. Finally, another period of time referred to as the horizontal back porch adds to the delay before a new line of video data is started.

Similarly, a vertical synch pulse is generated at the end of a visible frame of video, including a vertical front porch and a vertical back porch delay.



Part 2: Simulation of a simple VGA controller

a) Open the Project4.qar file in Quartus, and compile the design.

Question 1: How many registers are used in this design? _____

Question 2: How many pins are used in this design? _____

b) The three Verilog source files are found in the source subdirectory of the project.

Question 3: What is the frequency of the VGA_CTRL_CLK? (found in Project4.v)

_____

Review video_sync_generator.v file. cHD shows the calculation for when the horizontal synch pulse is active, and cVD shows the calculation for when the vertical synch pulse is active. Note that both synch pulses are active high in this design.

Question 4: When is the horizontal synch pulse active? Answer based on the h_pixels values.

_____

Question 5: When is the vertical synch pulse active? Answer based on the v_lines values.

_____

c) Create a testbench to simulate Project4. The input clock frequency should be 50Mhz, and the `timescale value should be 1 ns. The testbench must generate a clock, and drive the KEY[0] input low then high to reset the system. Set KEY[0] to 0 at time 0, and set KEY[0] to logic 1 at time 25.

Allow the testbench to run for 20,000,000 simulation time units (20 milliseconds). Running the simulation this long will generate a .vcd file size of about 50MB, so make sure you have disk space available.

Open the simulation in GTKwave. Add the following signals in the GTKwave display:
clock
KEY
VGA_HS
VGA_VS
VGA_R
VGA_G
VGA_B
DUT/U0/U0/h_cnt
DUT/U0/U0/v_cnt

Question 6: How many vertical synch pulses are shown in the simulation?

_____

Question 7: At what simulation time does VGA_VS go high?

_____

Question 8: When VGA_VS goes high, what is the v_cnt value in decimal?

_____

Question 9: What is the maximum value of v_cnt (in decimal) seen in this simulation?

_____

Question 10: What is the maximum value of h_cnt (in decimal) seen in this simulation?

_____

Question 11: Review the VGA outputs in the simulation, and then review the code in the vga_controller.v file. Describe the pattern this design would display on a monitor.

Take a screen capture of your testbench file. Paste the screen capture at the end of this document.

Take a screen capture of your GTKWave output waveforms. Zoom completely out so that the entire 20ms of simulation time can be seen. Paste the screen capture at the end of this document.

Part 3: What is Signal Tap and What Does Signal Tap Do?

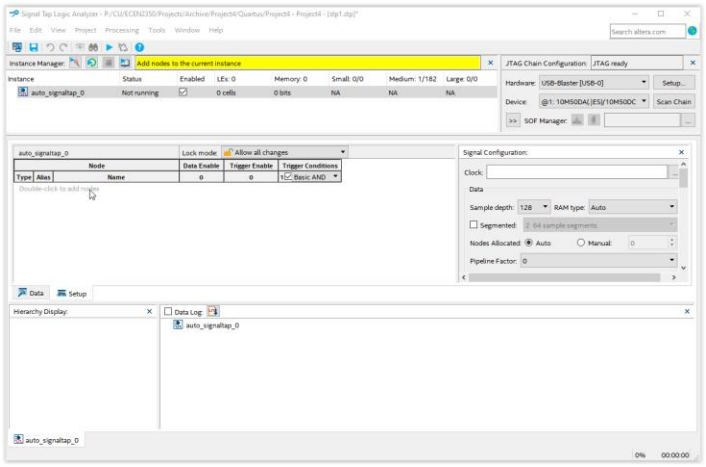
Quartus contains an important debug component called SignalTap. SignalTap is a logic analyzer tool constructed using FPGA lookup tables, registers, and memory. By adding a SignalTap analyzer to your design, signals internal to the FPGA can be captured real time, and displayed on your host computer. You can control what triggers logic capture, how many samples are stored, etc. Adding a SignalTap analyzer does not require the use of any additional pins, and provides a very capable tool that will aid circuit debug.

SignalTap captures data and stores the data into memory blocks of the FPGA. Once the capture memory is full, the memory contents are uploaded via the JTAG programming cable to the SignalTap GUI and displayed for evaluation. The data capture is done real time, the data

upload is done in non-real time.

Part 4: Add Signal Tap Analyzer to the VGA design

1) In your Quartus project, go to File > New and select Signal Tap Logic Analyzer File, and press OK. The Signal Tap window will open. The yellow bar at the top of screen says “Add nodes to the current instance”, so that is what we will do.



2) In the center left of the Signal Tap window you will see “Double-click to add nodes” (find the mouse pointer in the screenshot above). Double click and the Node Finder window will open.

3) You should see a Filter drop down in Options area of the Node Finder window. If you do not see this, click the blue box to the right of the List button. Set the Filter drop down to Signal Tap: pre-synthesis (you may need to scroll down to find this setting), and click the list button. A list of possible capture signals are shown.

4) Add these signals to the analyzer, by selecting the signals in the left hand pane labeled Matching Nodes:, and moving then over to the right hand pane labeled Nodes Found: using the arrows:

VGA_CTRL_CLK
VGA_HS, VGA_VS
VGA_R, VGA_G, VGA_B
vga_controller:u0/video_synch_generator:U0/h_cnt
vga_controller:u0/video_synch_generator:U0/v_cnt

Click the Insert button, then Close.

Note: SignalTap can be used to capture any combination of signals in your design. Step 5 selected the signals we are interested in. One of the drawbacks of a tool like SignalTap is that you must recompile your project whenever you add or remove signals to SignalTap.

5) In the Signal Configuration pane, select MAX10_CLK1_50 as the clock. Set the sample depth to 1K, Trigger position: to Center trigger position, and Trigger conditions: to 2. Do not change any other settings.

Note: We will use the 50Mhz clock as our sample clock. The data will trigger such that half the samples occur prior to the trigger, and half the samples occur after the trigger. We will create a trigger that allows two defined conditions to trigger. The capture memory will be 1024 samples deep.

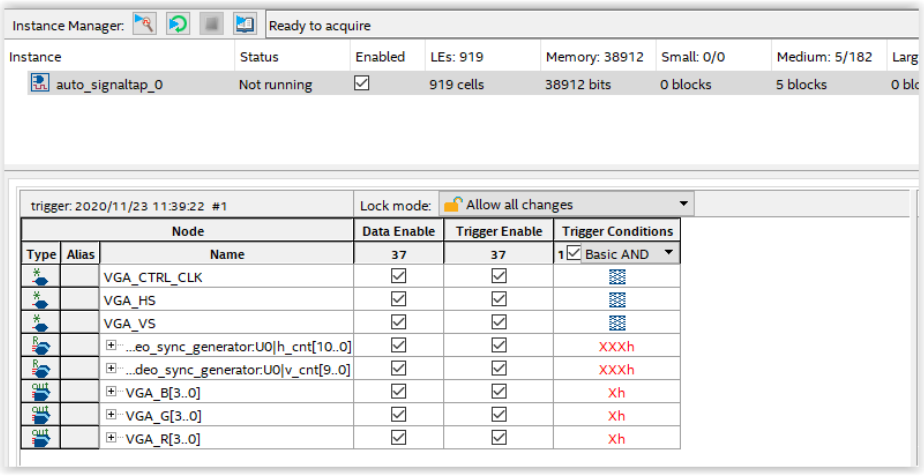
6) In the SignalTap GUI, go to Processing > Start Compilation. Select yes when asked to save your Signal Tap file, and accept the name provided. If prompted to add Signal Tap to your project, select yes.

7) In the Quartus GUI, when compilation is finished, notice that the number of logic elements and registers is much higher, and memory bits are used. This increase of logic in your design is due to the SignalTap core being included.

Part 5: Let's Capture VGA Data!!!

1) Connect your board to the USB programming cable. In the SignalTap GUI, in the upper right area labeled JTAG Chain Configuration, confirm that the Hardware: entry is showing USB-Blaster [USB-0], and that the Device: entry start with @1: 10M50……. Click the box with 3 dots and select the programming file Project4.sof. Click the blue download icon to program your board.

2) Your board is now programmed with the video controller containing the SignalTap analyzer.



The top pane now reads “Ready to acquire”, meaning that SignalTap is reading to be armed. Before arming SignalTap, we need to set the trigger conditions.

3) For the first test, let’s capture the value of the horizontal pixel counter as it switches from 799 to 0 (end of a line). In the center of the screen where your signals are listed, find h_cnt. In the Trigger conditions column, click in the Trigger Conditions column, right click and select Insert Value. Set the value as 799 decimal.

4) Capture data by selecting Processing > Run Analysis, or by pressing the Run Analysis icon, or by pressing F5. The auto_signaltap_0 instance must be highlighted for data capture to occur. You should see SignalTap indicate that data capture is occurring, and then see waveforms displayed.

5) Find the h_cnt row, and click in the waveform area to zoom in. In the waveform area, your left mouse button will zoom in and the right mouse button will zoom out. If the h_cnt values are not displayed in decimal, right click and set the radix to unsigned decimal.

Take a screen capture of the waveform section, when you have zoomed in enough to see the h_cnt value of 799 changing to 0. Paste the waveform at the end of this document.

6) Now let’s look at the line counter and vertical synch. Select the Setup tab (the Data tab is active during waveform display). In the VGA_VS row, right click in the Trigger Conditions column and select Rising Edge. Reset the h_cnt value to don’t cares. Rerun data acquisition. Change the v_cnt radix to unsigned decimal, and zoom so that you can see the value of v_cnt when VGA_VS goes high.

Take a screen capture of the waveform section, when you have zoomed in enough to

see the h_cnt value of 799. Paste the waveform at the end of this document.

7) Set the trigger value of h_cnt to 799 decimal, and the value of v_cnt to 524. Remove any other trigger conditions. Rerun SignalTap.

Take a screen capture of the waveform section, when you have zoomed in enough to see the h_cnt value of 799 and v_cnt value of 524. Paste the waveform at the end of this document.

8) Set the trigger values of h_cnt and v_cnt to don't cares. Set both VGA_HS and VGA_VS trigger conditions to rising edge. Capture data. What happens?

Question 12: What happened in question 8 above? Explain why this behavior occurred.

Grading

Project4 is worth 75 points total. Unlike the previous projects, your submission will be an updated version of this document containing the information requested in this document.

If you are unable to complete everything, submit what you have for partial credit

Note: You must submit Project 4 no later than 11:59pm MST on Monday, December 7th. No late submissions will be accepted.