EEL 4712C: Digital Design
Spring 2024

LAB 04: VGA & Timing Analysis
(50 Points)

Pre-Lab Objectives

- The goal of the first portion of this lab is to introduce you to the Quartus static timing analysis tool, TimeQuest, using the previous lab. Later in this lab TimeQuest will be used again.
- Parts 2 and 3 of this lab will require you to interface with the onboard VGA output first to display a fixed object, then to have the object move around the screen.
- In the final part of this lab, you will do part 1 again but this time in your VGA project. It is important to create the clock constraint before flashing your device because of the larger size of this design relative to prior labs.
- The demo of this lab will be over 2 weeks with parts 1 and 2 of the lab due the first week of this lab and parts 3 and 4 due the week after.

Pre-Lab Requirements

Week 1 Deliverables
Part 1: TimeQuest Timing Analysis (5 points)

To get started with the static timing analysis tool TimeQuest, open lab 3 in Quartus and use the following tutorial to generate a timing constraint file and slack reports for the design.

1. In the task window of your project, click the play button next to "Timing Analysis" to generate only the necessary output files for performing timing analysis.

2. As a result of the compilation, a compilation report will be generated. In the compilation report table of contents, open the folder titled 'Timing Analyzer' and open the tab titled 'Clocks.'

3. In the 'Clocks' window that appears, there should on be one item named 'clk'. Right click on 'clk' and click 'Report Timing… (In Timing Analyzer UI).'
   - This will bring up 2 windows, the timing analyzer and the smaller report timing dialog box. Under "from clock:" write 'clk' without quotes and hit report timing.
4. The previous step has now generated a timing report on all connections/paths starting at a register clocked using the 'clk' signal and ending in the same clock domain (i.e. paths between 2 registers where both registers use 'clk'). Take a screenshot of the output to include in your report.
   - For Lab 2, this is true of all paths, but it will not always be true of larger designs!

5. The report generated should show red on the timing of every path because, by default, TimeQuest uses a clock with a 1ns period. With this constraint, it is almost impossible to have data output from one register available to read by the next register in time. To create a constraint more accurate to the DE-10 Lite, go to the drop down menu Constraints → Remove Clock and select 'clk' from the drop down. Hit run.

6. Now a new clock constraint can be created using the 50MHz clock of the DE-10 Lite. Go to the dropdown Constraints → Create Clock. In the dialog box that appears type in a name and change the clock period to 20ns.

7. Next, click the '…' button next to Targets, click 'List', highlight the clock for the project, and move it to the right column with the > button. Choose 'OK' then 'Run.' The clock constraint has now been created. Close the timing analyzer and click yes when asked if you would like to save your constraint to a file.

8. Include your newly generated constraint file (.sdc file-type) in the project and rerun the timing analysis, reopen the report window and take a screenshot of the new slack times for your report. The times should now be positive numbers and no longer highlighted in red.

Note: If Lab 3 is still unfinished, you may use Lab 0 or Lab 2 to do this step but not Lab 1 as it does not have a clock input.

Part 2: Still Image on VGA (15 points)

There will be 3 main components of the VGA module created in this lab. You will need to modify your clock divider from Lab 0, create a component that generates the VGA synchronization signals, and create a VGA driver capable of parsing the outputs of the VGA Sync Gen to determine what to draw on the monitor.

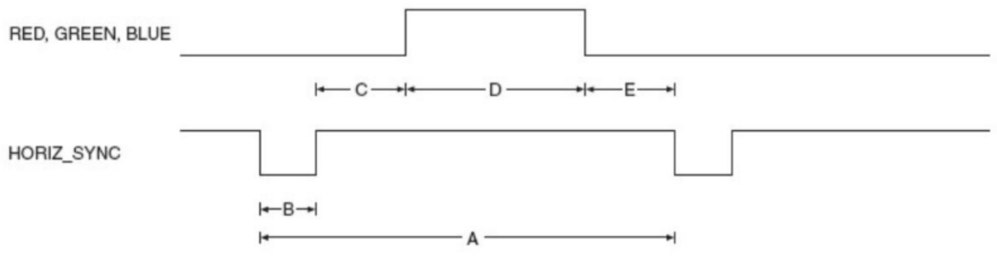
1. Using the clock divider implemented in Lab 0 and your knowledge of generic component inputs, change your clock divider to accept 2 generic inputs: input frequency, and output frequency.

    - As a refresher on how the clock divider works, the clock generator should count to some value that corresponds to the number of clock pulses from the input clock there should be in the output clock.

    - The output of this clock divider should be passed into the VGA entity.

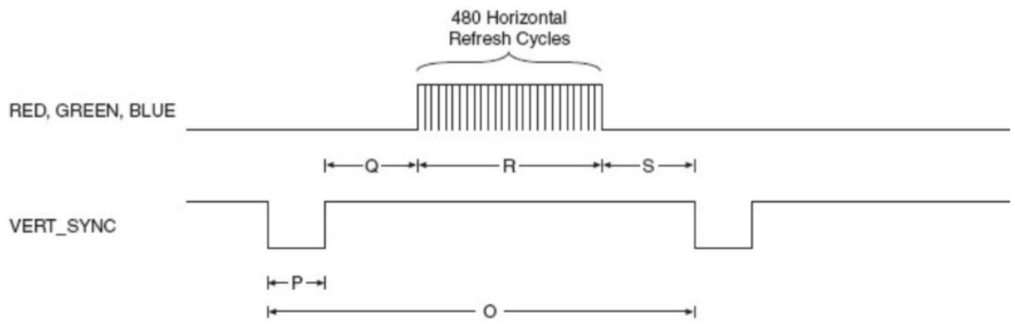
2. Using the provided vga_sync_gen entity, create a behavioral entity defining the behavior of the VGA synchronization signals as defined below.

- Create two counters with equal width outputs of COUNT_RANGE as found in the provided vga_lib.vhd file.
- h_count_r
  - Continually counts up to the horizontal period (H_MAX – See vga_lib.vhd) and then starts over at 0, using the 25 MHz pixel clock.
  - A value of zero on h_count_r corresponds to the beginning of section D in Figure 1.
- v_count_r
  - Counts up to the vertical period (V_MAX – See vga_lib.vhd). It will increment at a particular point in the horizontal counters count (Hcount = H_VERT_INC – See vga_lib.vhd).
  - A value of zero on v_count_r corresponds to the beginning of section R in Figure 2.
- The values of video_on, h_sync, and v_sync are determined by comparing the values of h_count_r and v_count_r with the constants provided in vga_lib.vhd.



Parameters	A	B	C	D	E
Time	31.77 μs	3.77 μs	1.89 μs	25.17 μs	0.94 μs

Figure 1. Horizontal Refresh Cycle.



Parameters	O	P	Q	R	S
Time	16.6 ms	64 μs	1.02 ms	15.25 ms	0.35 ms

Figure 2. Vertical Refresh Cycle.

3. To complete the remaining functionality of the VGA module, you will create an architecture for the provided VGA entity that instantiates the vga_sync_gen from the previous step and uses the outputs to draw a 64x64 pixel square in the center of the monitor with 12-bit hex value of #73b.

    - The VGA implementation partitions the screen into 2x2 pixel-sized blocks, each of which displays a color made from a combination or red, green, and blue. There will be a total of 4096 blocks arranged as a 64x64 grid, which forms a 128x128 image. VGA resolution is 640x480, so you must make sure the pixels not used by the image are black.

    - Note that the VGA entity has a video_on output, which is not part of a normal VGA interface. We are including it for verification purposes. The video_on signal should connect to the video_on output of the vga_sync_gen entity. Note in the top-level entity that the video_on output is left open because it is not used outside of the testbench.

    - To properly determine when to draw your image, you will need to implement two logical elements in your design.

        - **Row bound logic**: Use the v_count signal to determine the position currently being drawn on the monitor. Only enable your vertical output if the v_count value corresponds to the proper rows on screen.

        - **Column bound logic**: Use the h_count signal to determine the horizontal position currently being drawn on the monitor. Only enable your horizontal output if the h_count value corresponds to the proper columns on screen.

            - I recommend an enable signal that makes the color output all zero if both the row and column logic don't output one.

4. Run the provided VGA testbench to test the timing of your VGA outputs and adjust your timing as necessary until there are no errors remaining. This testbench will also output a .txt file that can be used to simulate the VGA monitor after verifying in the testbench. It is a fairly strict test for timing to make sure the outputs closely align with Figures 1, 2, & 3.

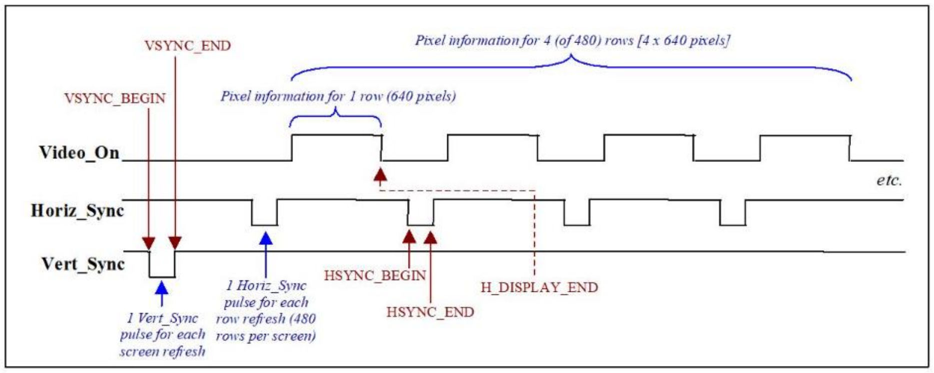

VSYNC_END
Pixel information for 4 (of 480) rows [4 x 640 pixels]
VSYNC_BEGIN
Pixel information for 1 row (640 pixels)
Video_On
Horiz_Sync
Vert_Sync
etc.
1 Horiz_Sync pulse for each row refresh (480 rows per screen)
HSYNC_BEGIN
H_DISPLAY_END
HSYNC_END
1 Vert_Sync pulse for each screen refresh

Figure 3. Timing Diagram for four rows of a VGA Display.

Week 2 Deliverables
Part 3: Display Moving Image on VGA (10 points)

The final task in this lab is to use the enable signal in the provided VGA entity to make the square drawn on the monitor start and stop moving. This enable signal is tied to switch 0 in the top level file.

1. Adjust the code from the previous part of this lab such that the square drawn on the monitor moves towards the edge of the screen while the enable is held high. When the square collides with the edge of the screen, it should change directions so that it bounces around the screen. When enable goes low again the square should stay in its current position.

Part 4: VGA Timing Constraints (5 points)

Follow the instructions again from part 1 of this lab, making sure once again to take screen shots, to add a timing constraint to the VGA project. As labs get more complicated, these steps should always be followed, even if not explicitly stated, to generate a timing constraint appropriate for the DE-10 Lite. Without the proper constraints the synthesis tool might improperly route your design on the silicon and cause severe timing problems.

Verification

To test your VGA entity, a thorough testbench has been provided. Before attempting to simulate your design make sure that the VGA entity produces no errors in this testbench otherwise it almost certainly will not work. Timing of synchronization signals is extremely important for the VGA output so incorrect timing is practically guaranteed not to work.

After checking the timing of your signals, the testbench will generate a .txt file that you can use with **this tool** made by previous Digital Design TA Benjamin Wheeler to simulate the output of your VGA module on a monitor. The tool is still a beta so if there are any issues please let me know and I will try my best to fix them.

Demo Procedure (15 points)

Week 1

1. Show your TA the timing report generated after creating a new timing constraint for the prior lab.
2. Using the provided top_level.vhd and the VGA monitors in lab, connect your board and show the square with correct color code.

Week 2

1. Show your TA the timing report generated after creating a new timing constraint for this lab.
2. Using the provided top_level.vhd and the VGA monitors in lab, connect your board and show the square bouncing around the screen.

Deliverables & Submission Guidelines

- Summarize your work in a report. Template for lab report is provide on Canvas
- Zip your lab solutions and report in a file name <Your_UFID>.zip with a folder structure as shown below.
    - Make sure to use the file names shown below in your submission

```
Your_UFID/
├── P1/
│   ├── no_constraint_report.jpg
│   ├── constraint_report.jpg
│   └── clk_constraint.sdc
├── P2/
│   ├── vga_tb_simulation.jpg
│   ├── monitor_simulaiton.jpg
│   ├── clk_divider.vhd
│   └── vga_sync_gen.vhd
├── P3/
│   ├── vga_tb_simulation.jpg
│   ├── monitor_simulaiton.jpg
│   └── vga.vhd
├── P4/
│   ├── no_constraint_report.jpg
│   ├── constraint_report.jpg
│   └── vga_constraint.sdc
│
```

Provided Entities

VGA Sync Gen Entity

```
use work.vga_lib.all;

entity vga_sync_gen is
    port (clk                      : in  std_logic;
          rst                      : in  std_logic;
          h_count, v_count         : out std_logic_vector(COUNT_RANGE);
          h_sync, v_sync, video_on : out std_logic);
end vga_sync_gen;
```

VGA Entity

```
use work.vga_lib.all;

entity vga is
    port (clk              : in  std_logic;
          rst              : in  std_logic;
          en               : in  std_logic;
          red, green, blue : out std_logic_vector(3 downto 0);
          h_sync, v_sync   : out std_logic;
          video_on         : out std_logic);
end vga;
```

Dr. Stitt’s notes on the functionality of VGA

The VGA monitor connector on the ALTERA DE10-lite board consists of five signals, RED(4 bits), GREEN(4 bits), BLUE(4 bits), HORIZ_SYNC, and VERT_SYNC. Look into the user manual for the header pin outs for these 5 signals. The timing relationships among this signal are shown in Figures 1 and 2 at the end of this lab document. The generation of signals needed for the raster on the VGA monitor begins with dividing the frequency of the 50 MHz clock on the ALTERA DE0 board down to a 25 MHz pixel clock which is further divided down to the horizontal and vertical sync frequencies. This means your design will contain a horizontal counter and a vertical counter. The horizontal and vertical sync pulses of appropriate lengths are then produced from the two counters.

Horizontal and vertical synchronization. A video display consists of 640 pixels in the horizontal direction and 480 lines of pixels in the vertical direction. The monitor starts each refresh cycle by updating the pixel in the top left-hand corner of the screen, which can be treated as the origin (0,0) of an X–Y plane. After the first pixel is refreshed, the monitor refreshes the remaining pixels in the row. When the monitor receives a pulse on the HORIZ_SYNC pin, it refreshes the next row of pixels. The time required for the sweep, the horizontal sweep period, is nominally 31.77 ms. This process is repeated until the monitor reaches the bottom of the screen. When the monitor reaches the bottom of the screen, a 64 ms pulse applied to the VERT_SYNC pin causes the monitor to begin refreshing pixels at the top of the screen (i.e., at [0,0]). As shown in Figure 2, the VERT_SYNC pulse must be repeated every 16.6 ms (vertical sweep period). A complete screen of information is traced by the electron beam every 16.6 ms for a refresh rate of 60 Hz.

Blanking intervals. In order to accommodate the time required to move the electron beam back to the left side of the screen (horizontal retrace period) the electron beam must be turned off during the retrace time or the return trace would be visible. This is accomplished by two blanking intervals during the horizontal refresh cycle marked by B and C at the beginning of the trace and E at the end of the trace in Figure 1. Similar blanking intervals are required during vertical refresh (P, Q, and S in Figure 2). The color beams are turned on only when the horizontal counter is in the period D and the vertical counter is in period R.

Hint: Blanking intervals can be easily implemented by creating a video “gate” signal (video_on) that is true only when the color beams can be active. Refer to the figure 3 at the end of this lab write-up for a graphical look at the VGA display signals.