# ECE 484 – Video Synchronization Laboratory Assignment

Last Updated: 1 March 2013

## Lab Overview

In this lab, you will write a VGA controller in VHDL and implement it on your FPGA development board. You will be provided a VGA-to-HDMI module that will automatically format your output for the HDMI output port on your development board. Your video controller will be written using a modified version of the major-minor FSM methodology learned in the lecture.

## VGA Overview

Video Graphics Array (VGA) is an interface protocol used to transmit analog video data to a screen. The VGA protocol uses a scanning method to project an image on the screen. Starting in the top-left of the screen, the monitor will progressively move from left to right, top to bottom to display each pixel. The following signals must be sent to a VGA monitor in order to display an image.

1. (red, green, blue) – three separate analog voltage signals indicating the amount of each color to display in the current pixel. These signals are sometimes abbreviated as RGB.
2. h\_sync – Horizontal synchronization signal that tells the screen to start writing pixels to the next line
3. v\_sync – Vertical synchronization signal that tells the screen that the current video frame is completed. The screen then starts writing pixels to the top-left of the screen.

Both synchronization signals contain four unique states: active\_video, front\_porch, sync\_pulse, and back\_porch. Incoming pixel data (through the RGB channels) is only displayed during the active\_video state of the synchronization signals.

Internally, **you will use a 25 MHz clock as your “pixel clock.”** On the rising edge of this clock, when both the h\_sync and v\_sync signals are in the active\_video state, you will place the RGB values you want the screen to display for that pixel. During all other states, the RGB values must be “0.”

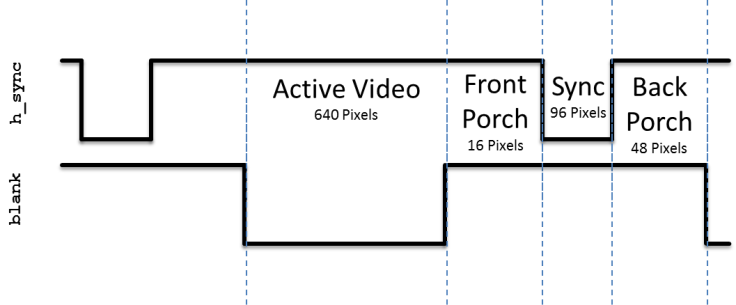


Figure : The h\_sync signal contains four states. Pixel data is only displayed on the monitor during the active\_video state. During all other state, the RGB values must be "0".

The v\_sync signal looks nearly identical to the h\_sync signal, however it is significantly stretched out in time. Where the h\_sync signal was counted in terms of pixels (based on the pixel clock), the v\_sync signal is counted based on iterations of the h\_sync signal. For example, in Figure 2, the active\_video portion is active for 480 complete iterations of the h\_sync signal.

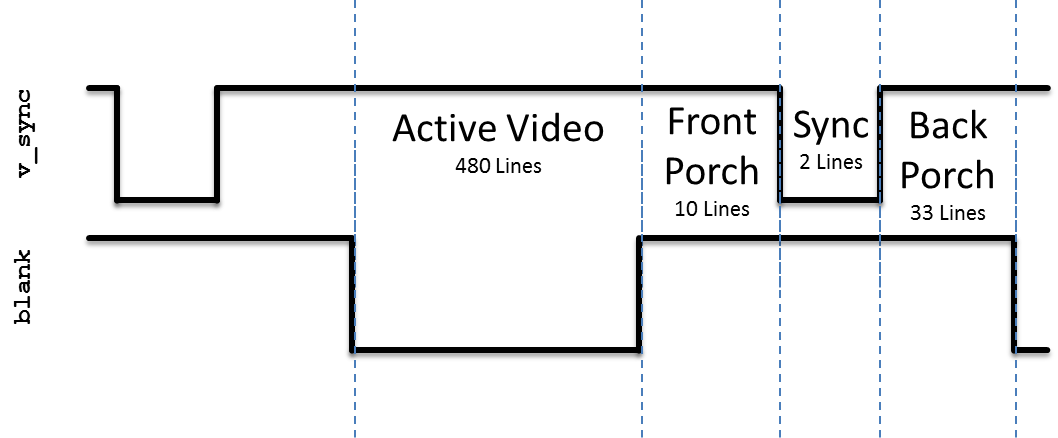


Figure : The v\_sync signal is similar to h\_sync, but instead of counting based on the pixel clock, the states are based on the number of iterations of the h\_sync signal. Pixel data is only displayed on the monitor during the active\_video state. During all other state, the RGB values must be "0".

More details on the VGA protocol can be found at <http://www-mtl.mit.edu/Courses/6.111/labkit/vga.shtml>. This link provides the exact numbers needed to generate the correct timing pulse signals for any VGA resolution.

## HDMI Overview

Generating DVID and HDMI signals is beyond the scope of this lab, but the following are excellent resources to learn the basics of these protocols:

1. Xilinx XAPP460 – Video Connectivity Using TMDS I/O in Spartan-3A FPGAs
2. Xilinx XAPP495 – Implementing a TMDS Video Interface in the Spartan-6 FPGA
3. Digital Visual Interface (DVI) Specification – Revision 1.0, 02 April 1999 – See pages 28 and 29 for TMDS encoding algorithm

## Prelab Assignment

On the first day of the lab, turn in a typed hard copy the following items:

1. Draw state-transition diagram for the h\_sync\_gen module which implements the timing diagram shown in Figure 1. Just consider count to be an input to your FSM.
2. Draw the conceptual diagram for count\_reg and count\_next signals based on "inputs" state\_reg and state\_next.
3. What is the Boolean expression for the completed signal? Note that this signal is high for exactly one clock cycle – the *last* clock cycle in the back porch state.

## VGA Hardware Implementation

To successfully complete this lab, you will need to implement the modules shown in on the next-to-last page of this document.

* h\_sync\_gen and v\_sync\_gen modules
  + Generate the needed VGA synchronization signals: rgb pixel color vector, h\_sync, and v\_sync.
  + Create simple blank signals to indicate when each module is not in the active video state.
  + Create row and column unsigned numbers to let the pixel generator know which pixel to write to the screen
  + Create a one-clock-cycle pulse, completed, during the *last* clock cycle of the back porch state.
  + Hint: The h\_completed signal is used to increment the count in the v\_sync\_gen module. You will use the v\_completed signal in the next laboratory assignment.
* vga\_sync module
  + Structurally connects the h\_sync\_gen and v\_sync\_gen modules
  + Generates a blank signal based on v\_blank and h\_blank. **This *overall* blank signal should be low when *both* h\_sync\_gen and v\_sync\_gen are in active video.**
* pixel\_gen module
  + Display a test pattern (see Figure 3) on the screen to aid you in testing the quality of your synchronization signals.
  + If any of the horizontal or vertical dividing lines are blurry or moving, you know that your signals are slightly out of sync.
* ece484\_lab\_video\_tld is an example top-level module that the instructor used to implement this lab assignment

The h\_sync\_gen completed signal will pulse high on the last clock cycle of the h\_sync signal to allow the v\_sync\_gen to count the number of lines completed.

The Atlys clock is 100 MHz. Since our pixel clock will run at 25 MHz, we will use the same 25 MHz clock to synchronize all the elements within our circuit. In addition, because of HDMI signal generation requirements, you must also generate a 125 MHz clock.

There are three code listings at the end of this document that will help you in this lab:

* Code Listing 1 – Template entity declarations for the modules you need to create in this lab. By using these templates, your code will be plug-and-play with others so you can test/debug if an individual part does not work. For example, once you finish your horizontal synchronization module, an instructor can plug your code into the solution, and check if your design works correctly in hardware.
* Code Listing 2 – Template for your top-level design. This is provided to you to show how the DVI/HDMI modules are connected to your VGA module.
* Code Listing 3 – The constraints file that connects your top-level design ports to the required pins on the FPGA.

## Lab Hints

* Use generics to define the size of active video, sync pulses, etc. This will be critical since you may need change the screen resolution in future labs.
* Use a package header to define global constants (e.g, size of active\_video, sync\_pulse; the states common to both FSMs, etc.)
* Use a look-ahead output buffer for your FSMs to prevent glitches
* Refer to your FPGA board’s reference manual for the pins needed for the video signals and global clock.
* **Watch out for inferred latches!** In the past, *all* hardware problems in this lab were caused by cadets ignoring this advice.
* If your simulation is acting “weird,” i.e. signals are not changing when they should, be sure you have all the appropriate signals in the sensitivity list.
* Only write out RGB values when blank is low. Otherwise you should put out “00000000” (black). Failure to do this will prevent the monitor from correctly synchronizing with your signal.

## Extra Credit

Add additional features to this lab for extra credit. Here are a few ideas, but you can come up with your own as well:

* Change the test pattern or screen resolution in real-time based on switch configuration
* Creating a moving “AF” logo on the screen.

## Laboratory Documentation

Your laboratory report must include the following:

* **Introduction** – Provide a brief overview of the problem.
* **Implementation** – Provide block-diagram of your solution and briefly describe how you implemented each the modules. Include your state-transition diagram. Commented VHDL code should be included in an appendix.
* **Test/Debug** – Briefly describe the methods used to verify system functionality. List the major problems you encountered and how you fixed them. This should cover all the problems you encountered in the lab and how you fixed them. It is highly encouraged that you present this section in bullet-format. At a minimum, break each problem and solution into separate paragraphs.
* **Conclusion** – Explain what your learned from this lab and what changes you would recommend in future years to this lab or the lectures leading up to this lab.

Additional report requirements:

* Use a mono-spaced font (e.g., Courier New, Consolas) for VHDL items (entity, signals, etc.).
* Avoid first-person pronouns
* Proofread your report for formatting and grammar.
* Ensure your code is well formatted. Code should be printed with syntax highlighting, correct indentation, comments, and no code lines should be wrapped on the next line.



Figure : Your pixel\_gen signal must create a test pattern similar to the one shown in the figure. This test pattern will aid in testing the quality of your synchronization signals.

|  |
| --- |
| **entity** h\_sync\_gen **is**  **port** **(** clk **:** **in** std\_logic**;**  reset **:** **in** std\_logic**;**  h\_sync **:** **out** std\_logic**;**  blank **:** **out** std\_logic**;**  completed **:** **out** std\_logic**;**  column **:** **out** unsigned**(**10 **downto** 0**)**  **);**  **end** h\_sync\_gen**;**  **entity** v\_sync\_gen **is**  **port** **(** clk **:** **in** std\_logic**;**  reset **:** **in** std\_logic**;**  h\_blank **:** **in** std\_logic**;**  h\_completed **:** **in** std\_logic**;**  v\_sync **:** **out** std\_logic**;**  blank **:** **out** std\_logic**;**  completed **:** **out** std\_logic**;**  row **:** **out** unsigned**(**10 **downto** 0**)**  **);**  **end** v\_sync\_gen**;**  **entity** vga\_sync **is**  **port** **(** clk **:** **in** std\_logic**;**  reset **:** **in** std\_logic**;**  h\_sync **:** **out** std\_logic**;**  v\_sync **:** **out** std\_logic**;**  v\_completed **:** **out** std\_logic**;**  blank **:** **out** std\_logic**;**  row **:** **out** unsigned**(**10 **downto** 0**);**  column **:** **out** unsigned**(**10 **downto** 0**)**  **);**  **end** vga\_sync**;**  **entity** pixel\_gen **is**  **port** **(** row **:** **in** unsigned**(**10 **downto** 0**);**  column **:** **in** unsigned**(**10 **downto** 0**);**  blank **:** **in** std\_logic**;**  rgb **:** **out** std\_logic\_vector**(**7 **downto** 0**));**  **end** pixel\_gen**;**  **entity** atlys\_lab\_video **is**  **port** **(**  clk **:** **in** std\_logic**;** -- 100 MHz  reset **:** **in** std\_logic**;**  tmds **:** **out** std\_logic\_vector**(**3 **downto** 0**);**  tmdsb **:** **out** std\_logic\_vector**(**3 **downto** 0**)**  **);**  **end** atlys\_lab\_video**;** |

Code Listing 1 - Entity templates for the lab to ensure consistency between student designs.

|  |
| --- |
| -- TODO: Include requied libraries and packages  -- Don't forget about `unisim` and its `vcomponents` package.  -- TODO: Entity declaration (as shown on previous page)  **architecture** your\_last\_name **of** atlys\_lab\_video **is**  -- TODO: Signals, as needed  **begin**  -- Clock divider - creates pixel clock from 100MHz clock  inst\_DCM\_pixel**:** DCM  **generic** **map(**  CLKFX\_MULTIPLY **=>** 2**,**  CLKFX\_DIVIDE **=>** 8**,**  CLK\_FEEDBACK **=>** "1X"  **)**  **port** **map(**  clkin **=>** clk**,**  rst **=>** reset**,**  clkfx **=>** pixel\_clk  **);**  -- Clock diveider - creates HDMI serial output clock  inst\_DCM\_serialize**:** DCM  **generic** **map(**  CLKFX\_MULTIPLY **=>** 10**,** -- 5x speed of pixel clock  CLKFX\_DIVIDE **=>** 8**,**  CLK\_FEEDBACK **=>** "1X"  **)**  **port** **map(**  clkin **=>** clk**,**  rst **=>** reset**,**  clkfx **=>** serialize\_clk**,**  clkfx180 **=>** serialize\_clk\_n  **);**  -- TODO: VGA component instantiation  -- TODO: Pixel generator component instantiation    -- Convert VGA signals to HDMI (actually, DVID ... but close enough)  inst\_dvid**:** **entity** work**.**dvid  **port** **map(**  clk **=>** serialize\_clk**,**  clk\_n **=>** serialize\_clk\_n**,**  clk\_pixel **=>** pixel\_clk**,**  red\_p **=>** red**,**  green\_p **=>** green**,**  blue\_p **=>** blue**,**  blank **=>** blank**,**  hsync **=>** h\_sync**,**  vsync **=>** v\_sync**,**  -- outputs to TMDS drivers  red\_s **=>** red\_s**,**  green\_s **=>** green\_s**,**  blue\_s **=>** blue\_s**,**  clock\_s **=>** clock\_s  **);**  -- Output the HDMI data on differential signalling pins  OBUFDS\_blue **:** OBUFDS **port** **map**  **(** O **=>** TMDS**(**0**),** OB **=>** TMDSB**(**0**),** I **=>** blue\_s **);**  OBUFDS\_red **:** OBUFDS **port** **map**  **(** O **=>** TMDS**(**1**),** OB **=>** TMDSB**(**1**),** I **=>** green\_s **);**  OBUFDS\_green **:** OBUFDS **port** **map**  **(** O **=>** TMDS**(**2**),** OB **=>** TMDSB**(**2**),** I **=>** red\_s **);**  OBUFDS\_clock **:** OBUFDS **port** **map**  **(** O **=>** TMDS**(**3**),** OB **=>** TMDSB**(**3**),** I **=>** clock\_s **);**  **end** your\_last\_name**;** |

Code Listing - Template for your top-level design. You must place the correct code to replace the "TODO" comments.

|  |
| --- |
| VCCAUX = 3.3;  NET "clk" LOC = "L15" | PERIOD = 100 MHz;  NET "reset" LOC = "F5" | IOSTANDARD = LVCMOS33;  NET "TMDS(0)" LOC = "D8" | IOSTANDARD = TMDS\_33 ; # Blue  NET "TMDSB(0)" LOC = "C8" | IOSTANDARD = TMDS\_33 ;  NET "TMDS(1)" LOC = "C7" | IOSTANDARD = TMDS\_33 ; # Red  NET "TMDSB(1)" LOC = "A7" | IOSTANDARD = TMDS\_33 ;  NET "TMDS(2)" LOC = "B8" | IOSTANDARD = TMDS\_33 ; # Green  NET "TMDSB(2)" LOC = "A8" | IOSTANDARD = TMDS\_33 ;  NET "TMDS(3)" LOC = "B6" | IOSTANDARD = TMDS\_33 ; # Clock  NET "TMDSB(3)" LOC = "A6" | IOSTANDARD = TMDS\_33 ; |

Code Listing 3 - Required UCF Contents

|  |  |
| --- | --- |
| Video Synchronization Cut Sheet | **Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Instructor:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Section:** \_\_\_\_\_\_\_\_ |

**Number of hours spent on this lab:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (no points associated with this unless you leave it blank)

**Suggestions to improve this lab in future years:** (use blank space below)