ECEN 3002 – Video Project, Phase One

Introduction

To truly learn FPGA design, a person must spend time doing design, running the FPGA tools, simulating the design, and recognizing the proper debug techniques. The purpose of this course is not to make you an expert digital designer, but to introduce you to the key aspects of FPGA design, and how to take best advantage of what is offered.

The first goal of the video project is to create a simple VGA based video controller. Each week you will add additional features and capabilities to the project, as a means to learn more about available design and implementation options and approaches.

For Phase One, you should complete the following:

1) Create a VGA display controller running at 640 x 480, 60Hz resolution

2) Create accurate simulations of the video timing

3) Interface the video controller to the buttons and switches of your DE10-Standard board such that you can animate a simple display and control the colors.

Specific requirements for Phase One are give later in this document, and are highlighted in red.

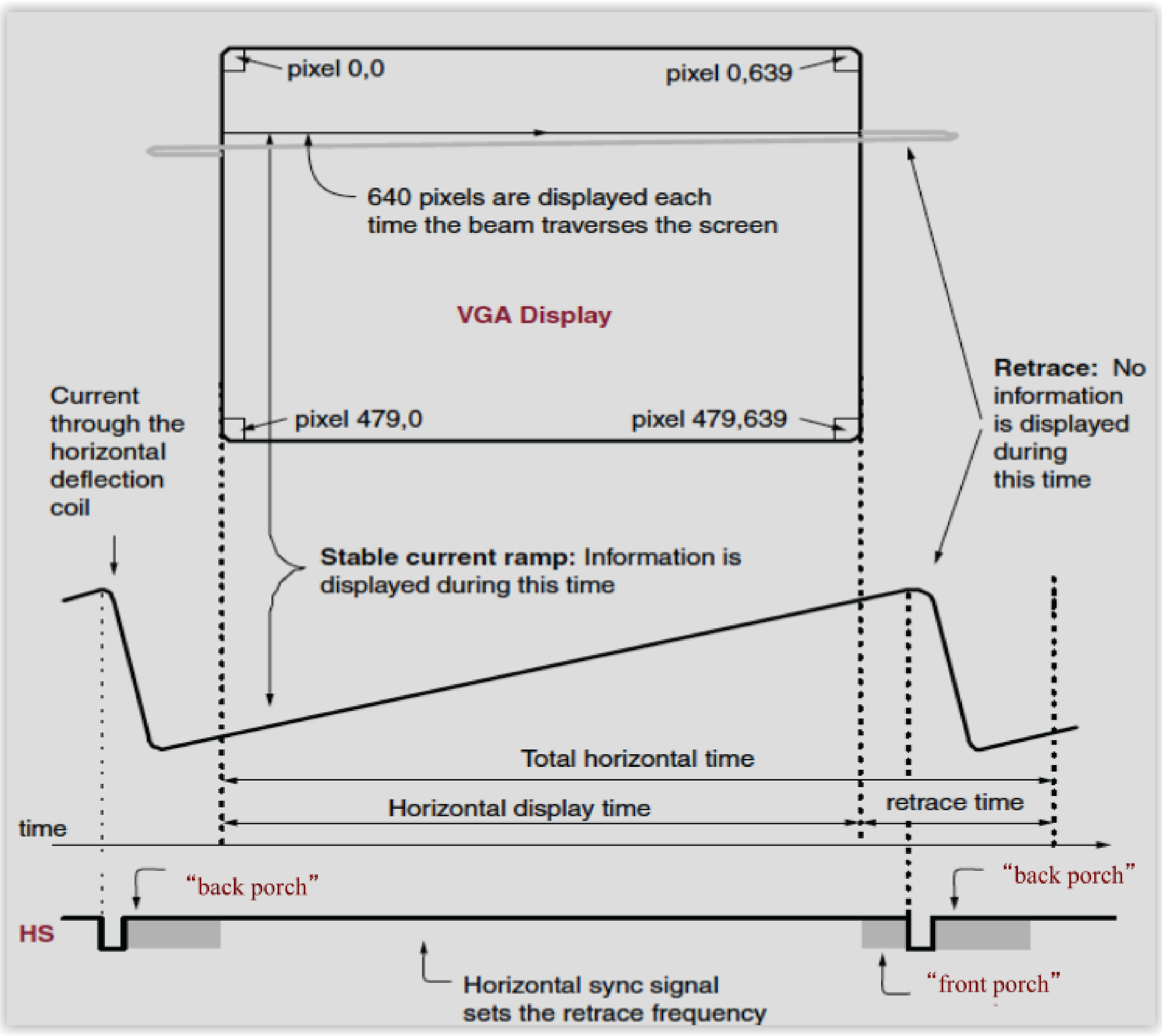
Basics of VGA video

The VGA video standard was developed for use with analog monitors. An electron beam scans across the monitor screen from left to right, and also moves from top to bottom at a given rate. At points along the beam travel, red, green, or blue values are painted on the screen phosphor. By controlling when the various colors are enabled, images are formed.

As the beam moves from left to right, eventually the beam comes to the right side of the screen, and must be moved or reset back to the left side. The Horizontal Sync (hSync) signal forces the beam back to the left side of the screen. Similarly, eventually the beam reaches the bottom of the screen, and the Vertical Sync (vSync) signal is used to move the beam back to the top of the screen.

A video controller generates the proper timing for the hSync and vSync signals. By using counters to horizontal and vertical beam position, one can control both when active video should occur, and when video should be disabled.

The diagram below provides detail for a 640 x 480 VGA controller.

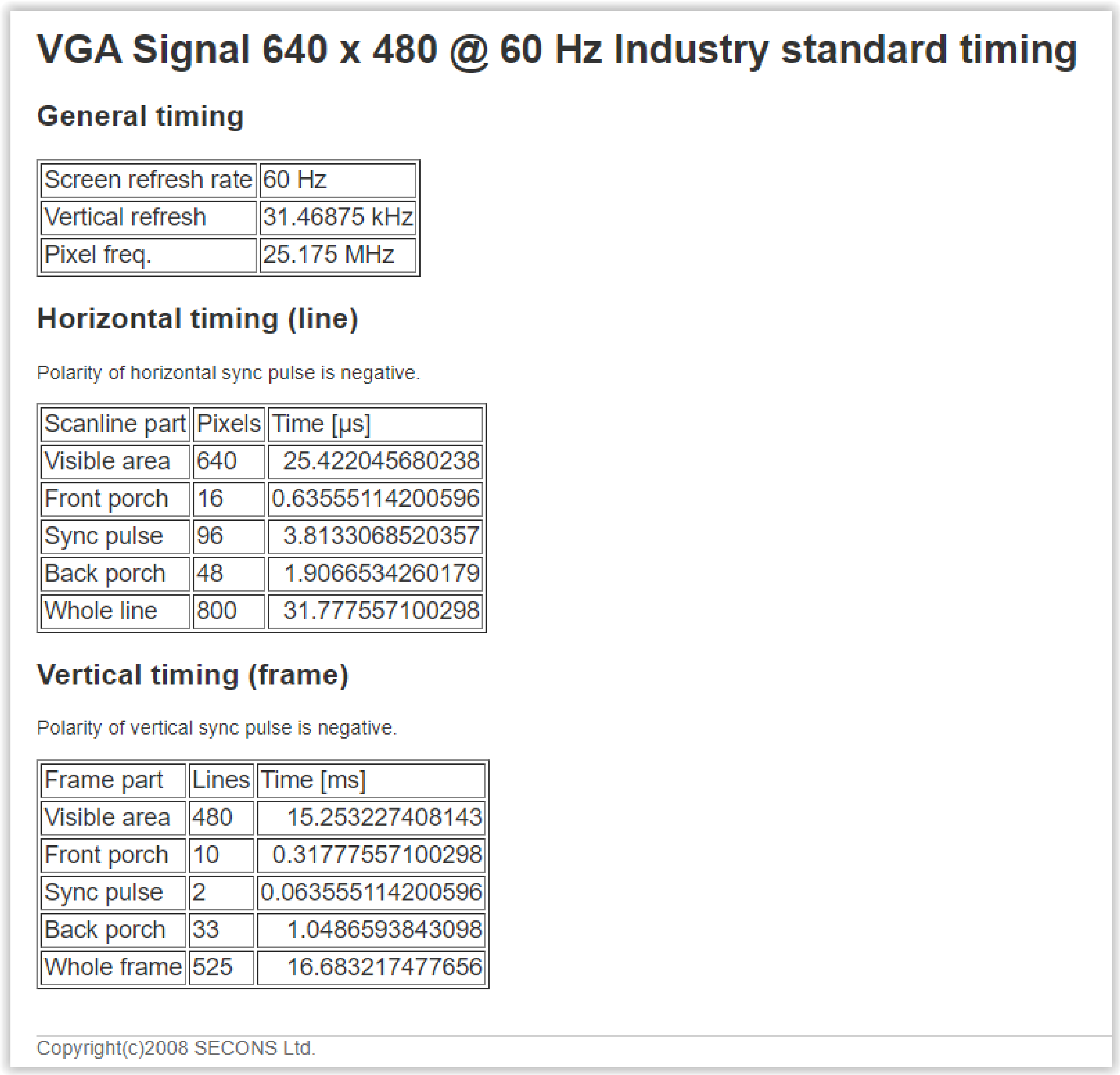

pixel 0,0
pixel 0,639
640 pixels are displayed each time the beam traverses the screen
VGA Display
Retrace: No information is displayed during this time
Current through the horizontal deflection coil
pixel 479,0
pixel 479,639
Stable current ramp: Information is displayed during this time
Total horizontal time
Horizontal display time
retrace time
time
"back porch"
"back porch"
HS
Horizontal sync signal sets the retrace frequency
"front porch"

Notice that there is time between hSync signals where there is no active video. Video is not active during the time when hSync is asserted low, as well as during a time period before hSync is active (called the front porch) and during a time period after hSync deasserts (called the back porch). This implies that the video controller must generate appropriate signals to indicate when active video signals must be driven, and when active video must be turned off. Time when video is off is termed the retrace time.

The same general discussion is also true for the vSync timing. An active low Vertical Sync pulse, along with its corresponding front and back porch intervals, signals the return of the beam to the top of the screen, and during this retrace period video must be turned off. The frequency of the vSync signal is called the refresh rate.

For 640 x 48, 60Hz VGA, the critical timing values look like this:

http://www.tinyvga.com/vga-timing/640x480@60Hz



VGA Signal 640 x 480 @ 60 Hz Industry standard timing

General timing

Screen refresh rate	60 Hz
Vertical refresh	31.46875 kHz
Pixel freq.	25.175 MHz

Horizontal timing (line)

Polarity of horizontal sync pulse is negative.

Scanline part	Pixels	Time [µs]
Visible area	640	25.422045680238
Front porch	16	0.63555114200596
Sync pulse	96	3.8133068520357
Back porch	48	1.9066534260179
Whole line	800	31.777557100298

Vertical timing (frame)

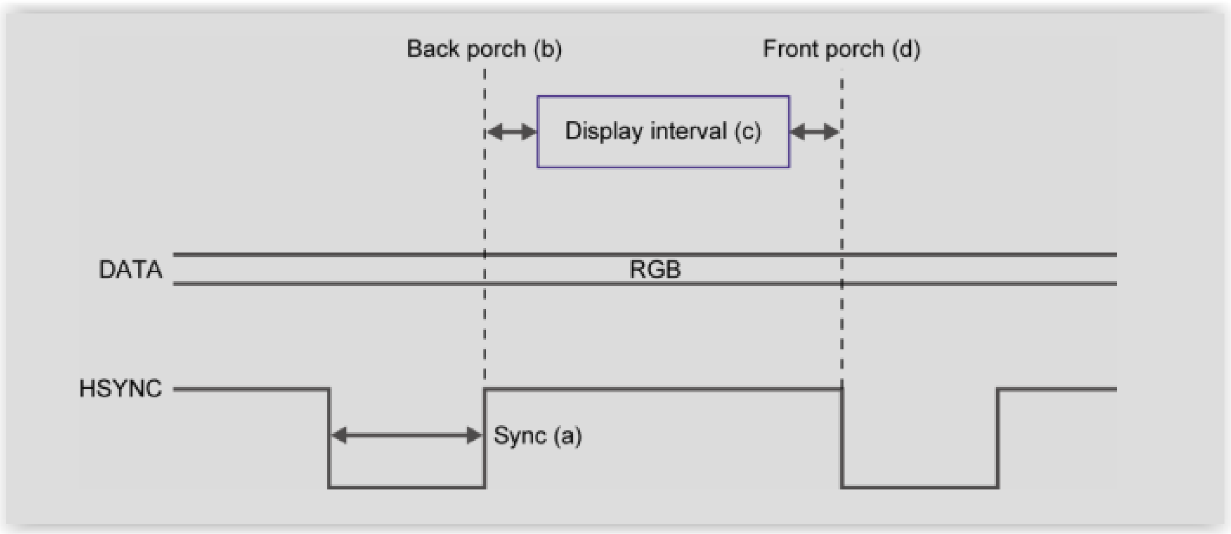
Polarity of vertical sync pulse is negative.

Frame part	Lines	Time [ms]
Visible area	480	15.253227408143
Front porch	10	0.31777557100298
Sync pulse	2	0.063555114200596
Back porch	33	1.0486593843098
Whole frame	525	16.683217477656

It is usually easier to think about timings in terms of pixels and lines, instead of time. A complete horizontal line consists of 800 pixels, of which only 640 pixels are actually displayed. Similarly, a complete screen consists of 525 lines, of which only 480 lines are actually displayed.

The diagram below shows the timing relationships for hSync.

To be entirely correct, a hSync pulse marks the end of a line and the beginning of the retrace period. However, when designing the controller, you may prefer to think as Hsync marking the beginning of a video line. As long as you get the timing right, the video will display correctly on the monitor.

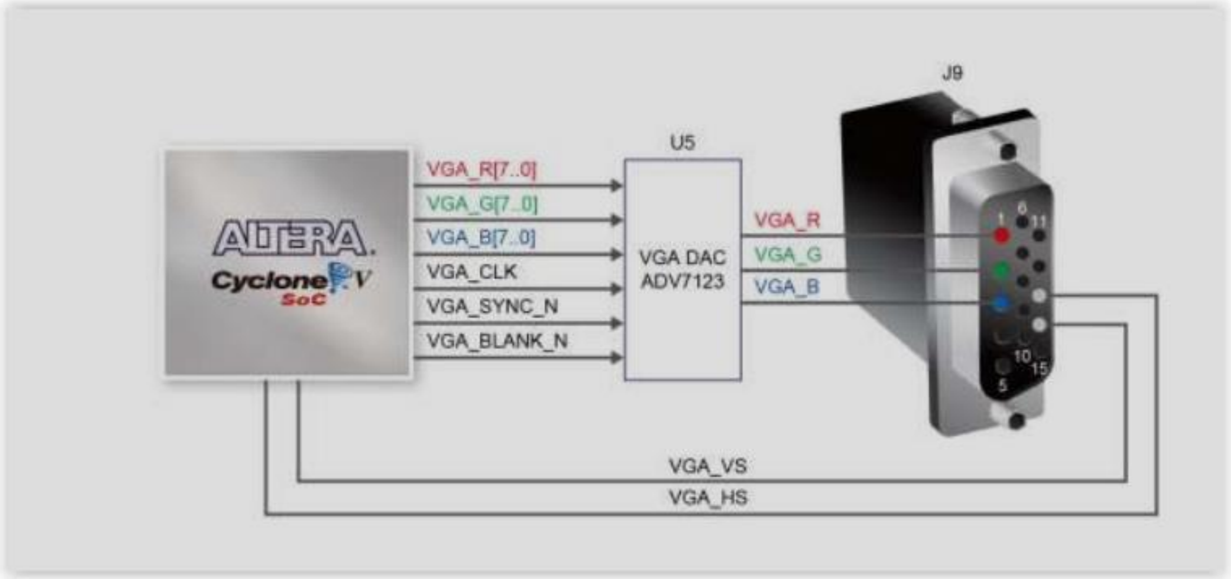


You can also use the timing diagram above for vSync, recognizing that vSync pulses signal the end of a complete frame of video, and that the display interval is the entire period where 480 lines of video are written.

At this point I think you have enough information to get started, except that we haven't dealt with how one turns on or off the active video.

VGA and Video DACs

The connections between the Cyclone V FPGA and the 15 pin VGA connector are shown in this diagram taken from the DE10-Standard user's manual.



The box labeled ADV7123 is a video DAC (Digital to Analog Convertor) made by Analog Devices. The datasheet for this DAC is contained in the DE10-Standard documentation package. By writing 8 bit values for each of the red, green, and blue (RGB) pixel values, the DAC will convert that information into the correct analog voltages and apply them to the red, green, and blue connector pins.

When the RGB inputs to the ADV7123 are driven with all logic 0s, video is turned off (0 v corresponds to a black screen).

For this lab, your design will drive the VGA_HS (hSync), VGA_VS (vSync), and the VGA_RGB signals. The VGA_BLANK_N signal should be tied high, and the VGA_SYNC_N signal should be tied low.

General Design Guidelines

1. Keep all your design fully synchronous.

2. Use a PLL to generate your video clock. Later in the project we will replace the PLL with something a little different, so put your clocking logic in a separate module to make this later transition easier.

If you are not familiar with how to generate and use a PLL, see the section titled PLLs later in this document.

3. All counters and/or state machine logic must have resets. Not only is this good design practice, but will allow your simulations to work. Synchronize the deassertion of reset.

4. Do not hard code video parameters into your Verilog. Write your code in a manner that will allow easy conversion between VGA resolutions. Place system video parameters in a separate header file, and reference the header file where needed. I will provide you with a document titled Managing Verilog Parameters for Synthesis and Simulation that describes how to you can store parameter values in a header file and reference those values as needed.

5. Separate your design into a Video Timing Controller block, and a Pixel Generator.

Design

1. Video Timing Controller (VTC)

You VTC should generate the hSync and vSync signals, as well out provide access to the horizontal pixel location and current line value. Also, create an active video signal that is asserted when in the active video region.

Construct your VTC using VGA 640 x 480, 60Hz timing values. When you have completed this block, construct a simulation that demonstrates correct operation.

If you are at a point where you can connect the DE10-Standard board to a monitor, you can verify at least basic operation of this circuit by driving a constant RGB value to the screen.

2. Pixel Generator (PG)

The purpose of the PG is to determine what color should be displayed on the screen at a given pixel location. Inputs to the PG include the horizontal and vertical locations from the VTC, and also the active video signal from the VTC. Other required inputs to the PG determine what color pixels to render at the given locations, which will be discussed next.

The outputs of the PG are the RGB values sent to the video DAC.

For this lab, create a simple "Compute on the fly" design that displays two boxes (see the Creating Video to Display section below). Use the SW inputs to modify the colors of each box. Use one or more pushbutton to cause the boxes to move on the screen.

Create a simulation that demonstrates the behavior or the PG. It is not necessary to simulate movement of the boxes if you don't want to.

The PG logic will be enhanced over the course of the semester.

Creating Video to Display

1. **Compute on the fly**

You can construct simple shapes and things on the screen simply by setting your RGB values based on your horizontal and vertical location.

For example, say you wanted to draw a vertical red line whenever your horizontal position was between 100 and 200. Your PG simply needs to look at the horizontal position counter, and set the RGB value to red whenever the counter is in the desired range. If you include the vertical counter in your calculation, you could draw a box.

Compute on the fly video creation is simple, but limited. It requires no video memory, but is very limited in terms of what can be done.

2. **Frame buffer**

A frame buffer is a block of memory that can be programmed with RBG values for each pixel. High performance systems often have two complete frame buffers so that one buffer can be modified while the other buffer is supply the video content. Single frame buffer systems often use a dual port memory approach. One port will be read by the PG to write data to the display, while the other port can be written by a CPU or debug system.

Frame buffer systems are more complex, and often strain FPGA memory capacities. For a 640 x 480 resolution, you need 307,200 locations. If you use the full 8 bits each for red, green, and blue that the DE10-Standard board supports, you will need 921,600 bytes of memory, more than our Cyclone V device contains.

You can use memory external to the FPGA, such as the SDRAM chip on the DE10-Standard board. Use of external memory has its own issues, typically performance related. Successful creation of a frame buffer using external memory requires careful design techniques to guarantee adequate memory bandwidth. Video is very unforgiving when the video data is not available.

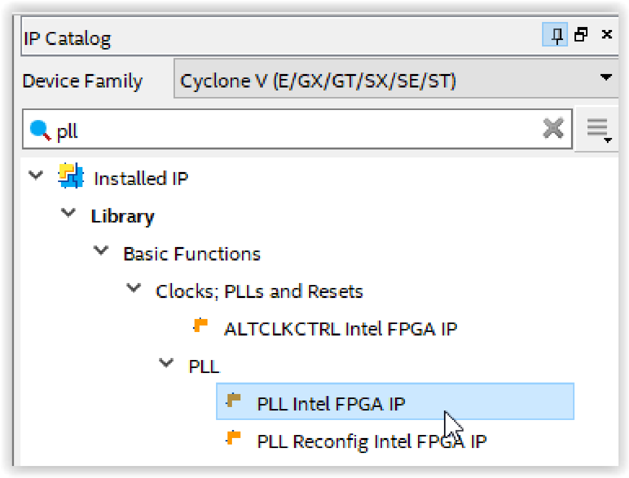
3. **Character mapping**

Since a full frame buffer is very memory intensive, there are numerous techniques to allow display of characters (or small game images like asteroids), and these techniques require much less memory. You divide the screen into small sections (for example 8 x 8 pixels), and then access prestored characters or icons / images from memory and display the appropriate pixels. Since the characters only require 8 bits each, and if you have a limited number of characters available, you can display a reasonable character / icon set using under 10Kbytes of memory.

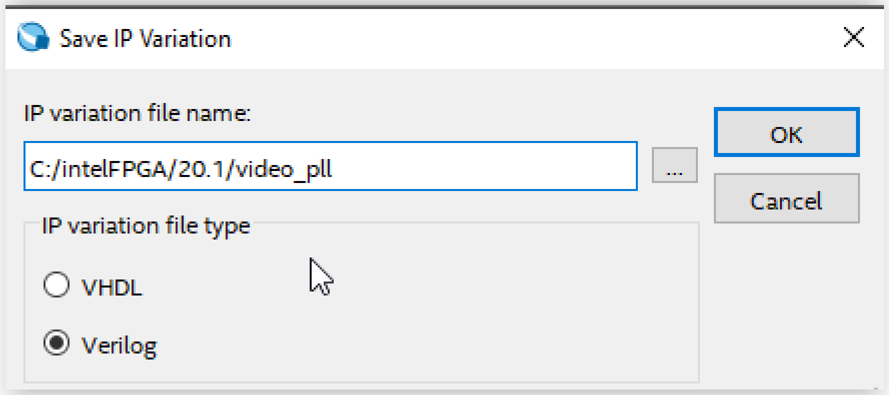
PLLs

A PLL (Phase Lock Loop) is a clocking component that can be used to create clocks of arbitrary frequency, duty cycle, and phase shift. The Cyclone V FPGA family has built in PLLs that can be accessed through the Quartus IP catalog. For this lab we will use a PLL to create a custom clock frequency for our VGA controller design.

Start by opening your project in Quartus, and go to the IP catalog on the right side of the screen. If you don't see the IP Catalog pane, open it by View > Utility Windows > IP Catalog. In the IP Catalog search for PLL, and double click on PLL Intel FPGA IP.


IP Catalog
Device Family: Cyclone V (E/GX/GT/SX/SE/ST)
pll
- Installed IP
  - Library
    - Basic Functions
      - Clocks; PLLs and Resets
        - ALTCLKCTRL Intel FPGA IP
        - PLL
          - PLL Intel FPGA IP
          - PLL Reconfig Intel FPGA IP

In the Save IP Variation window, provide a name of the PLL, make sure Verilog is selected, and click OK.



Save IP Variation

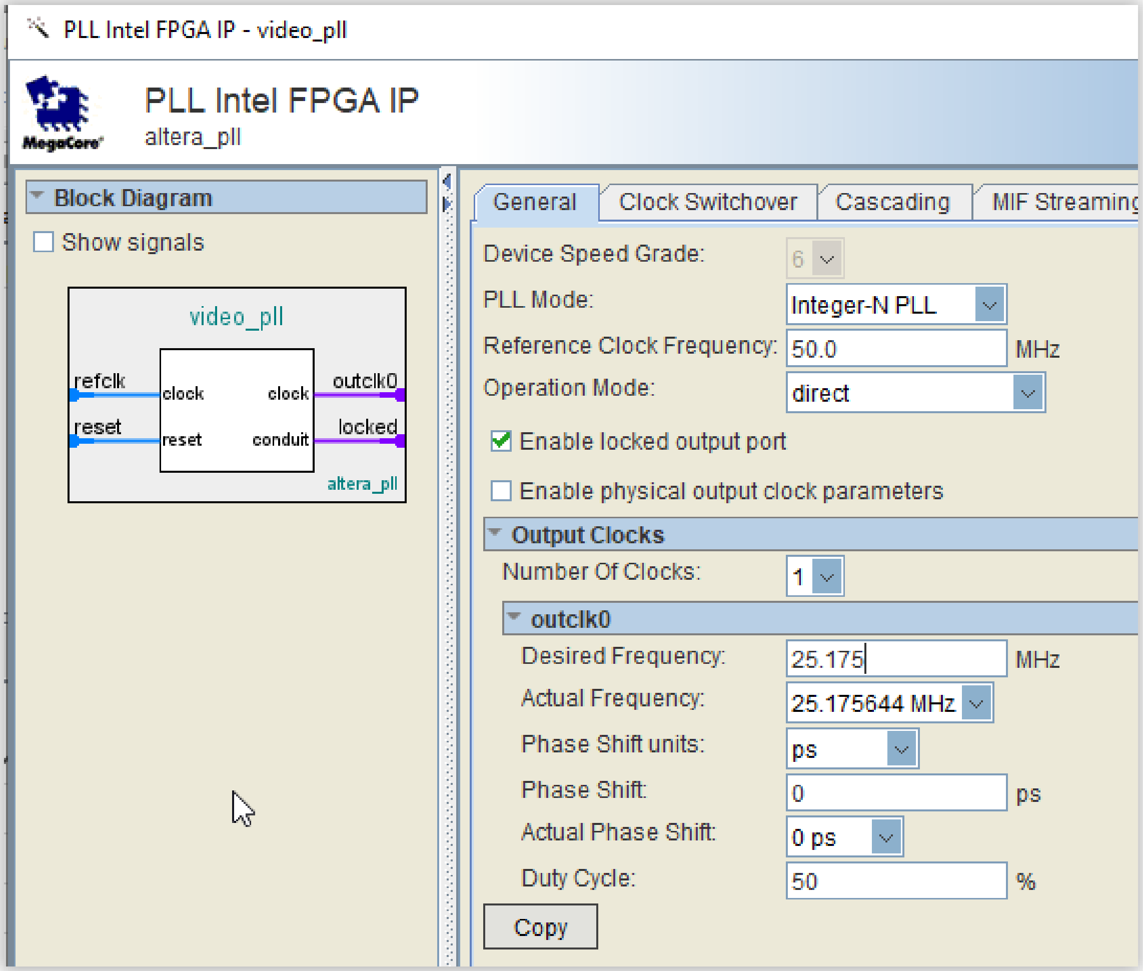
IP variation file name: C:/intelFPGA/20.1/video_pll

IP variation file type

( ) VHDL

(●) Verilog

When the PLL configuration dialog opens, in the General tab, change the Reference Clock Frequency to 50 Mhz, and the outclk0 Desired Frequency to 25.175 Mhz. Notice that the actual frequency reported will be slightly different, that will not be a problem. Click Finish.



PLL Intel FPGA IP - video_pll

PLL Intel FPGA IP
altera_pll

Device Speed Grade: 6

PLL Mode: Integer-N PLL

Reference Clock Frequency: 50.0 MHz

Operation Mode: direct

[✓] Enable locked output port

[ ] Enable physical output clock parameters

Output Clocks

Number Of Clocks: 1

outclk0

Desired Frequency: 25.175 MHz

Actual Frequency: 25.175644 MHz

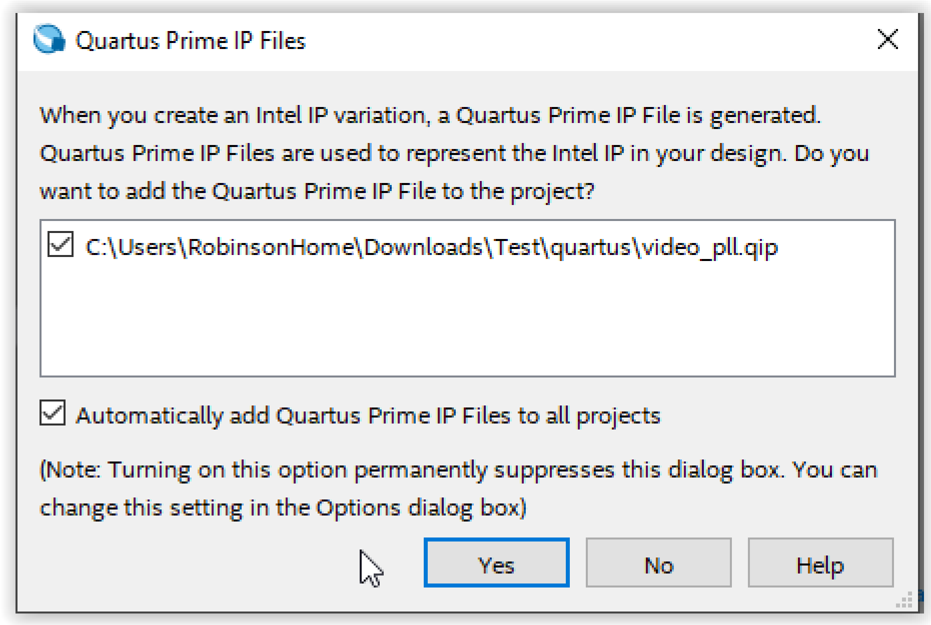
Phase Shift units: ps

Phase Shift: 0 ps

Actual Phase Shift: 0 ps

Duty Cycle: 50 %

The PLL IP will now be generated. When the window below opens, you can optionally select "Automatically add Quartus Prime Project Files to all projects", then click Yes.


Quartus Prime IP Files

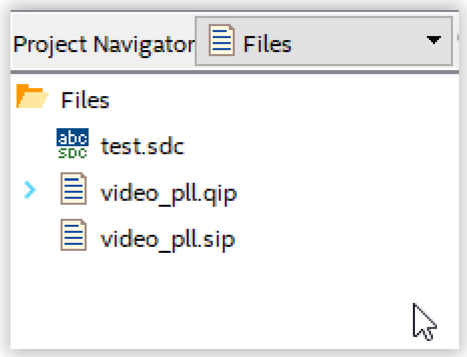
When you create an Intel IP variation, a Quartus Prime IP File is generated. Quartus Prime IP Files are used to represent the Intel IP in your design. Do you want to add the Quartus Prime IP File to the project?

☑ C:\Users\RobinsonHome\Downloads\Test\quartus\video_pll.qip

☑ Automatically add Quartus Prime IP Files to all projects

(Note: Turning on this option permanently suppresses this dialog box. You can change this setting in the Options dialog box)

Change the drop down in the Project Navigator pane to Files, and you will see the PLL files added to your project



In order to use this PLL IP in your design, expand the files under video_pll.qip, and open the file video_pll.v. At the top of the file you will see the instantiation template for the PLL.

```
module video_pll (
     input  wire  refclk,   //  refclk.clk
     input  wire  rst,      //   reset.reset
     output wire  outclk_0, // outclk0.clk
     output wire  locked    //  locked.export
     );
```

The `rst` polarity is positive high. The `locked` output indicates when outclk_0 is valid, as PLLs do take some time after power on to create a stable output clock.

Credits

Portions of this lab are based on work done by Professor Eric Brunvand and Vikas Rao at the University of Utah, as well as the Introduction to Video using FPGA apnote from Intel Corporation, authored by H. Martinez and A. Arenas.