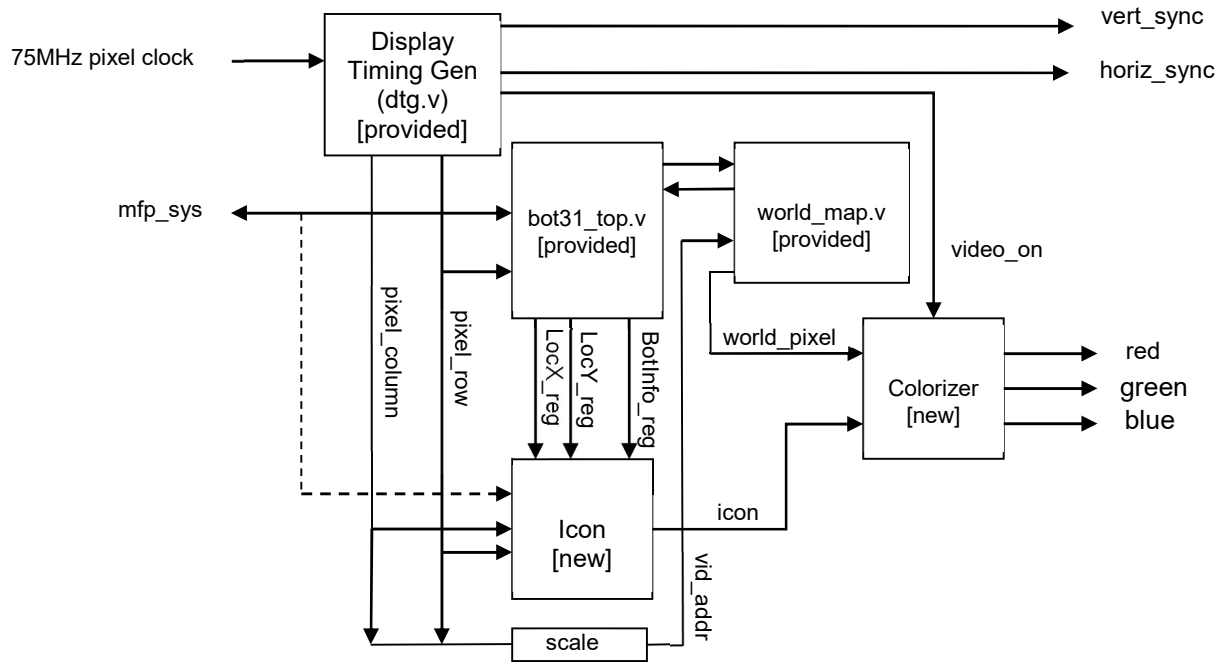
Rojobot World Video Controller

One of the primary goals for Project #2 is to give you experience designing and implementing video. For this part of the project you will implement an animated and colorful graphical display for your Rojobot world that displays an image of the Rojobot virtual world with the Rojobot moving around on it. Many project teams have built on what they learned implementing the video controller to make their final project "visually interesting" (one of the requirements of the final project). It is important to note that even though your output will be displayed on a VGA-compatible display, the controller you produce for this project is not a VGA graphics controller such as you would see on a PC or laptop.

The video controller for this project overlays an icon of your Rojobot (be creative here) onto a background generated from the world map. The icon moves through the Rojobot world based on the current{X, Y} coordinates of the Rojobot. These coordinates are returned by the Rojobot emulation in the LocX and LocY registers as described in the *Rojobot31 Functional Specification*. The orientation (which way the Rojobot is facing) is returned in the BotInfo register. This means that your Rojobot icon must clearly indicate which way it is facing.

Display Subsystem Architecture



A high-level block diagram for the display subsystem is shown above. The subsystem combines two bit maps to produce the image on the display. The Display Timing Generator (dtg.v) produces the pixel row and pixel column addresses. These addresses can be combined and driven to the port B (the world maps are implemented as a dual-port ROM) address input to the ROM. The resulting pixel stream is sent to the Colorizer to draw an image of the world map. The 128 x 128 world is displayed as a 1024 x 768 image on the VGA monitor so your implementation should be scaled to represent each world location as an 8 x 6

pixel block on the display. The two-bit `world_pixel` output indicates "open space", "obstruction", "black line", or "reserved" for each location on the screen.

The icon module stores a small (16 x 16) image of the Rojobot. The position of the Rojobot icon on the screen is determined by the values of the `LocX_reg` and `LocY_reg` outputs from the Rojobot. The icon module produces a two-bit output that indicates "transparent" or one of 3 opaque colors. The Colorizer module combines these two pixel steams into a 12-bit, 4096-color output. The Rojobot icon will appear in the foreground in front of the world background (e.g. the icon overlays the world map background at the Rojobot's location).

Display Timing Generator

The Display Timing Generator (DTG) generates the video raster signals Vertical Sync (`vert_sync`), Horizontal Sync (`horiz_sync`), and `video_on` which indicates the viewable region of the video screen; and `pixel_row` and `pixel_column`, which indicate the current vertical and horizontal pixel position on the display. Refer to the Nexys4 DDR Reference Manual for details.

The display image has 768 rows with 1024 pixels (columns) in each row. Each pixel consists of three colors: Red, Green, and Blue. The Nexys4 DDR supports 4096 colors (4 color bits per pixel for red, green, and blue for a total of 12-bit color). The pixel information is delivered to the monitor serially by row. The end of each row is indicated by a *horizontal sync pulse* of a particular length. At the end of all 768 rows, a longer, *vertical sync pulse* occurs. At the end of a horizontal sync pulse, the beam goes back to pixel 0 of the next row. At the end of a vertical sync pulse, the beam goes to pixel {0, 0} (the top left pixel on the display). The entire image is drawn on the display 60 times per second.

Icon Module

The icon module stores a 16 x 16 x 2 bitmap image of the Rojobot in read-only memory. This may be synthesized from HDL using a case statement. Alternately, you can use the Memory generator in the Vivado IP Integrator to produce one or more ROMs with the icon image pointing in different directions. Each pixel in the image is stored as two bits in the ROM. If the bits are '00' it means 'transparent', that is the background color (world map locations) will show through. Values 01, 10, and 11 means display one of the 3 colors of the Rojobot image; you can choose the colors. Thus the icon need not be rectangular, and can have windows in it. If a 16 x 16 pixel icon appears too small for your satisfaction, then scale the image up for improved appearance. Show each pixel as a 2 x 2 or 4 x 4 block on the screen.

Two 10-bit inputs determine the vertical and horizontal position of the top-left corner of the icon. The icon module compares the values in these registers with the current row/column display pixel position generated by the DTG to determine whether or not to display the icon at the current raster position. The X and Y position input values are compared with the DTG row/column values. Thus, with the proper scaling between the Rojobot's location (in Loc_X and Loc_Y), the icon can be positioned anywhere on the screen with a resolution of one pixel, vertically or horizontally. As the Rojobot icon moves through the world map the icon is drawn at the Rojobot's current location.

The icon module should also show the Rojobot's orientation. The `BotInfo_reg` output provides eight headings: 0, 45, 90, 135, 180, 225, 270, and 315 degree encoded into 3 bits [0 (0) – 315 (7)]. Design the icon logic to change the orientation of the icon image in response to the `BotInfo_reg` input. An efficient implementation might use two bitmaps, one for 0, 90, 180, 270 degrees, and another for 45, 135, 225, 315 degrees, but you are free to use 8 icons (one per orientation) if you prefer. The Verilog code can be written

to flip the image vertically and horizontally by changing the way the icon ROM row and column pixels are addressed. Draw pictures, it will help.

Colorizer Module

The Colorizer module maps the two pixel streams from the Rojobot and Icon modules to 12-bit colors on the display such that the Rojobot icon appears in the foreground while the world image forms the background. Its function is defined in the following table:

World[1:0]	Icon[1:0]	Color
00	00	Background
01	00	Black Line
10	00	Obstruction
11	00	Reserved
x	01	Icon color 1
x	10	Icon color 2
x	11	Icon color 3

The colors are 12-bit values of your choice – four bits each representing red, green and blue.

Make sure that the colorizer output is 000 (black) during the blanking interval, when the `video_on` signal from the DTG is zero.

MCCM (The Series 7 clock manager)

The 1024 x 768 VGA timing is based on a 75 MHz pixel clock. Use the Clocking Wizard in the Vivado IP Integrator (Flow Manager/IP Catalog/FPGA Features and Design/Clocking Wizard) to generate a 75MHz clock from the 100MHz oscillator input. It is fairly straightforward to navigate through the tabs in the Clocking Wizard to create one or more clock outputs. Use the 75 Mhz clock for the VGA controller modules and Rojobot, leaving the MIPSfpga running at a slower speed to avoid timing failures. We recommend that you use the Clocking Wizard to generate a 50MHz clock for the system logic and a 75MHz clock to drive the VGA logic and Rojobot, but you are welcome to experiment with different frequency combinations. The Clocking Wizard, coupled with the Series 7 MCCM and PLL circuitry provides synchronized clocks at a variety of frequencies both faster and slower than the 100MHz oscillator input. The clock routing and MCCM and PLL clock generators provide are described in 7-Series FPGAs Clocking Resources (UG472).

The Clocking wizard will generate a number of files including VHDL and Verilog instantiation templates. Like your other Verilog modules, the MMCM or PLL created by the Clocking Wizard must be instantiated in your top level module. Navigate to the IP Sources tab in the Project manager and expand the clock generator instance. Open the .veo file and cut/paste the instantiation into your top level module. This is the same thing you did for Project #1 except now the clock generator has two output clocks.

Given

The Project 2 release package includes several files to help you complete your display system design. Use these in addition to the Verilog you've already implemented for the project to complete the assignment. A description of the files is provided in *ECE 540 Project 2 List of Files*. We provide the dtg.v which generates

the pixel row and column address and the horizontal and vertical synch pulses. The rest of the VGA logic is yours to design, implement, and debug.

Project Tasks

- Design and implement the Colorizer module.
- Design and implement the Icon module.
- Generate and integrate the 75MHz clock using the IP Integrator Clocking Wizard.
- Integrate the Display Timing Generator, Icon, and Colorizer modules into your system.
- Add VGA signal output ports to `mfp_nexys4_ddr.v` and uncomment the VGA connector signals in the constraint file (`mfp_nexys4_ddr.xdc`)
- Check/fix your Project 2 application program to make sure it successfully traverses *world_map_loop*. This world map is more difficult to traverse than world_map but still consists of only right turns.

NOTE: You will need to replace the `world_map.ngc` file from the first part of the project with one of the other maps. The easiest way to do this is to delete the current `world_map.ngc` file from your project (right click on the file in the Hierarchy pane and Remove File from Project). Than Add the new `world_map.ngc` to the project in the same way you added all of the other files. You may leave `world_map.v` alone; it does not change.

- Check that your black line following algorithm correctly handles a right and left turns. *world_map_lr* is the map that you will use in your demo. You will need to replace the loop virtual world with the new one. The new virtual world should be named world_map like the others.

References

[1] *Digilent Nexys4 DDR Board Reference Manual,* Digilent Inc.
[2] *Xilinx Vivado Software Manuals*
[3] ECE 540 Project 2 write-up
[4] *Rojobot31 Functional Specification*, by Roy Kravitz
[5] *Rojobot31 Theory of Operation*, by Roy Kravitz
[6] *ECE 540 Project 2 List of Files*
[7] *7-Series FPGAs Clocking Resources* (UG472)