Marcin Wisniowski

Lab 3 – VGA Bouncing Ball

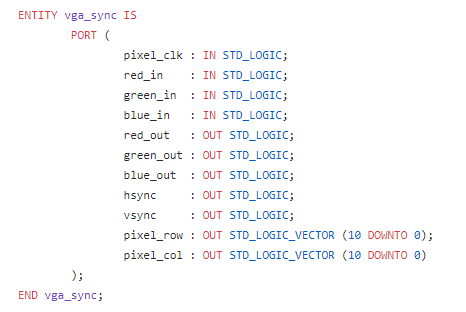
In this lab, the Nexys4 DDR board was programmed to allow for a bouncing ball to move across a monitor by using the VGA Display port that is available on the board. While the VGA format has been extended over the years to higher resolutions, a 1056x628 RGB display format is used during this lab as seen in the vga\_sync file.

The VGA protocol was initially designed to drive a cathode ray tube (CRT) that continuously updated horizontal and vertical lines. Each of these intersections displayed a pixel on the screen of which a red, green, and blue bit of information was assigned to define the brightness and color of that pixel. The screen is refreshed continuously with a frame rate that generates horizontal and vertical synchronization signals to scan the display and make all the changes in the pixels.

A horizontal HSYNC pulse triggers the horizontal scanning of the next line and a vertical VSYNC pulses the beam back on top. In this case, you can think of each as a trigger in a big table to go to the next row or column.

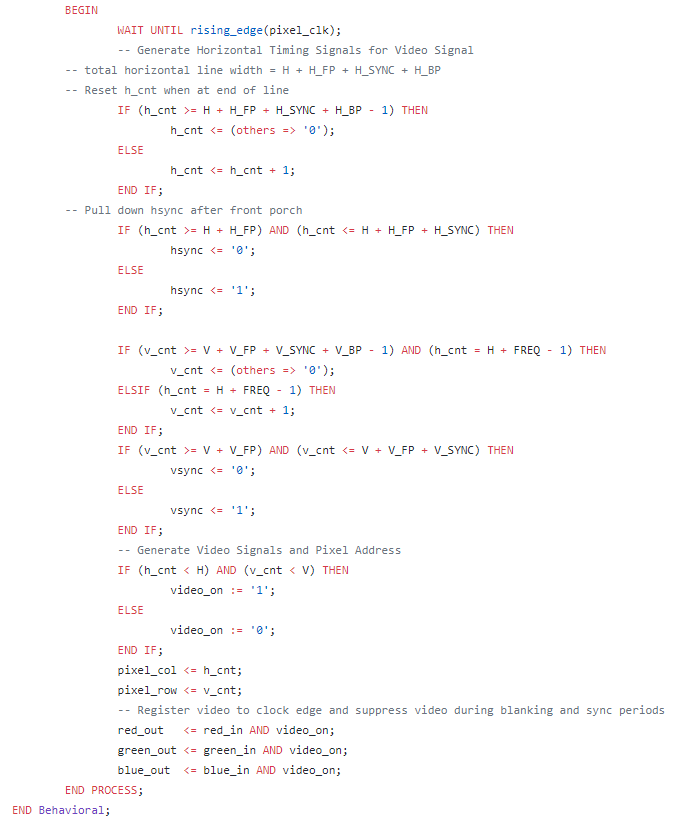
To start the coding, the vga\_sync file needs to be made in order to write the middleware to allow for the vga to correctly display given the FPGA design.

The vga entity is created with a pixel\_clk and desired pixel values as inputs and outputs of red/green/blue and correct locations to loop through.

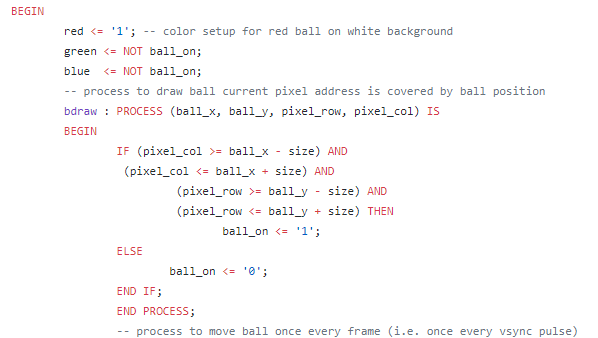


While the HSYNC and VSYNC triggers do the updating and scanning of the entire screen, the VGA port does not know what value to change the pixels to at each pixel node. The sync, therefore, pulses a red, green, and blue analog signal between 0V – 0.7V generated by the board’s video signal and a resistor digital to analog converter. Interestingly enough, there are less bits of blue intensity as your eye is less sensitive to smaller changes in blue levels and therefore visuals may look worse when in darker situations.

The vga\_sync code uses the FPGA clock to drive horizontal and vertical counters to generate the correct pixel addresses. There is a section that is outside the scope of the video settings called H\_SYNC and V\_SYNC which, as described above, gives the signal a bit of time to reset back to the beginning location.

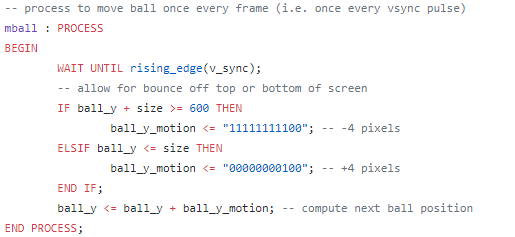


Once the firmware is written, a ball\_vga file can be created that references the vga\_sync code to create a ball object. In this section of the code, the ball’s pixel address is processed and the correct colors and motion speed is given to the object. In this code, the ball is written to be red and the motion is described by masking the pixels with a “ball\_on” property which changes the color of the pixel



What can also be seen above is the definition of the ball’s size. Here, the program describes in an if statement a square area around the ball that signifies the size and sets all of those values to be “ball\_on”. The ball\_x and ball\_y variables describe the current position of the ball while the pixel\_row and pixel\_col describe a generic pixel location. All pixel locations are looped through in the VGA\_sync file.

Afterwards, a process is created that gives a derivative of the motion (ball\_y\_motion) that describes the instantaneous speed of the ball’s movement. If the ball reaches the edges of the screens, here given as either 600 or 0 + size of ball, the ball\_y\_motion changes directions



Finally, a vga\_top file needs to be created to incorporate all the things that have been made. The vga\_sync and ball.vhd files can be thought of as classes that define the processes but do not actually run them. In order to run them, we use vga\_top to create the correct porting constraints from the FPGA pins.

