**VGA Display**

The files that we control the VGA display are *hsync\_generator.v, nexys3.ucf,* and, *CandyCrush\_VGA.v*. The first file synchronizes the vertical and horizontal positions on the display, defines the display area to be 480x640, and generates two counters to keep track of the x and y positions. This file was largely unchanged from the demo file provided to us.

The UCF file contains the interface specifications and timing constraints. To increase the number of different colors we were able to display, we initialized all eight pins on the VGA port; 3 pins for red and green, and 2 pins for blue. This allowed us to display up to 265 different 8-bit colors.

Lastly, *CandyCrush\_VGA.v* contains the actual logic we used to display a randomly-generated 8x8 gird of 8 colors. We chose 50x50 for the size of a square. From that, we stored the x and y intervals corresponding to each square in two arrays (one for x, one for y). That way, we could just use Boolean arguments to designate which square(s) we wanted the red/green/blue pins to signal. From this point, we struggled in figuring out the most efficient way to create the grid. Ideally, we could loop through the array of 64 random numbers between 1 and 8, assign each number to a color, and build the grid that way. Our issue was that we wanted to display all the colors at the same time, which requires one statement at the end of the logic to set each pin. The looping algorithm would reset the pins after each square was displayed, erasing the previous square. Ultimately, we had to simply hard-code the 64 square dimensions into each color pin. We used the randomly generated Grid[][] in Boolean expressions within these hard-coded pin statements to turn on/off the bits corresponding to the color we wanted to display.

While our method to display the grid was not the most code-efficient, we successfully created a random grid of 8 different colors.