- Instantiate entities (smaller boxes) in the bigger box
- Final output is 12 bits of RGB, Hsync, Vsync
    - Measure H and V up with timing at end of document
- Only part of screen will have image, rest is black
- Image is made of 64 blocks x 64 blocks
- Each block made up of 2x2 pixels
- Adjacent pixels have same RGB value
- 50 Mhz clock connected to pin G21
- Connects to clock divider which steps down to 25 Mhz
- Pixel clock connects to everything else in the top level
- Clock is not clock on rising edge (to go from 50 to 25)
- VGA sync gen entity
    - Takes in pixe clock as input, as well as reset
    - Outputs Hcount, Vcount, video on, Hsync, Vsync
    - Image is 640x480
    - Hcount, Vcount used to determine where you are on the screen
        - Hcount:
            - Implement as register
            - Put inside a process
            - On rising clock edge
            - if Hcount = Hmax (constant)
            - Reset Hcount back to 0
            - Else, increment Hcount
            - Increments on every rising edge
        - Vcount
            - Everything happns on rising edge
            - If Vcount = Vmax, reset Vcount to zero
            - Check to see if Hcount is equal to certain value. When it's equal to that value, increment Vcount
            - Else, hold Vcount
            - Implement as register, will automatically hold value
    - Hsync, Vsync are final outputs of entity
    - Video on basically is 1 most of the time when outputting valid RGB data
        - Goes to 0 on blanking intervals
    - Create internal signals so we can read Hcount and Vcount to get video on, Hsync, and Vsync. Use combinational logic to take in the counts and output the video on and syncs
    - Inside a process using the counts in the sensitivity list, default video on to 1 (want on most of the time)
        - Video on goes back to zero if Hcounnt > Hdisplay or Vcount > Vdisplay
    - Hsync, Vsync
        - Code in bottom left of picture
- Take Hcount and Vcount and decode them to get the correct address for the ROM
- Every Hcount corresponds to a column address, Vcount to row address
- Hcount tells where you are horizonallty

- Vcount tells you where you are vertically
- ROM
    - Has 12-bit address bus going in
    - 12-bit data going out (RGB data)
    - Has a clock
    - Every address in the ROM corresponds to RGB data
    - Every block has its own address in ROM (64x64 addresses)
    - Need 12 bits to display the value of 64x64
    - Column address is 6-bits
    - Concatenate row address w/ column address to get final ROM address which is what goes in the ROM
- How do we decode Hcount to get column address? (Baiscally column decoder)
    - Code on left of picture
    - Inside a process (Hcount, buttons) <= sens list
    - Going into column decoder, you have Hcount and buttons
    - Default column address and enable to zero
    - Case button (which button is pressed)
        - Center config
            - Check some things
    - Want to be able to normalize Hcount to zero
        - Code on left center of picture (Col addr = (Hcount – center_x_start)/2. . .)
- Column enable
    - Goes to 1 whenever you are in bounds
    - Code on left of picture
        - When left, do this
        - When ___, do that
- Row decoder/enable very similar, replace x with y, etc.
- Column enable & row enable tells us whether we are painting the image that is stored in ROM
- CE, RE, and video on (from vga sync gen)
    - AND all these together inside top level entitity
    - Goes into the select of a mux (implement with if else or whatever)
    - The output of the rom (RGB data) goes into the mux
    - Zero represents black
    - When CE, RE, and video on are 1, display data
    - When one of those signals is not one, want to display black or in one of the blanking intervals
-