COL 215 Hardware Assignment 2

Abhinav Shripad (2022CS11596)
Aneeket Yadav (2022CS11116)

October 2023

1 Problem Statement

1. To generate RAM and ROM using Vivado's memory generator.
2. To populate the ROM with a .coe file containing the grayscale pixel values.
3. To read each pixel value from the ROM, perform filter logic on it using sliding window logic and to store the processed pixel value in the RAM.
4. To read the processed pixel value from RAM and display the complete image on a VGA monitor.

2 Device Interface

The I/O ports of `entity device` are as follows:

- `clk : in std_logic`
- `r : out std_logic_vector(3 downto 0) ;`
- `g : out std_logic_vector(3 downto 0) ;`
- `b : out std_logic_vector(3 downto 0) ;`

This design choice has been taken so that sequential operations can be seemlessly managed using processes rather than declaring entities which are instead managed using internal signals Sequential VHDL is well-suited for modeling synchronous systems, where operations are synchronized to a clock signal. This is important in digital circuits to ensure that changes happen predictably and avoid issues like metastability.However care has to be taken in that the various processes are collectively concurrent.

2.1 Architecture

1. **Component declaration**:

   - `my_rom` : `dist_mem_gen_0`,
   - `my_ram` : `dist_mem_gen_1`

2. **Signal Declaration**

   Most of the variables in the code are fairly straightforward except perhaps for the below:

   - `signal ram_address_when_calc : std_logic_vector(15 downto 0) := (others => '0');`

     Represents the RAM address when pixels are being processed to calculate the gradient.

   - `signal ram_address_when_disp : std_logic_vector(15 downto 0) := (others => '0';`

     Represents the RAM address when pixels are being displayed on the VGA.

   - `signal init_result : integer := 0;`

     Represents the initial result obtained by processing three consecutive pixels.

   - `signal final_result : integer := 0;`

     Represents the result which has been clamped to 0 if `init_result` is negative and to 255 if `init_result` is greater than 255.

3. **Processes**

   An important observation is that processes corresponding to actions before displaying the pixel values on the VGA are synchronised with the 100MHz onboard `BASYS3` clock (`clk`) whereas those corresponding to later actions are synchronised with the 25MHz `clk_divided`.

   Non-trivial processes have been descibed below:

   - `adjust_ram_address`

     Used to assign ram_address depending on value of `write_enable`. This ensures that the correct `ram_adrress` is correctly accessed depending on whether the processed pixel values are being stored in the RAM or being displayed on the VGA.

   - `compute_gradient`

     Explanation behind using the mathematical specification of gradient:

     Calculates `init_result` for depending on the number of pixels traversed.

     (a) If (`i mod 256 /= 0 or i mod 256 /= 1 or`) i.e the sliding window consisting of three pixels does not extend beyond the ends of the grid.In this case, processed pixels value are calculated as per the formula `init_result <= pixel_1_value - 2 * pixel_2_value + pixel_3_value`

(b) If (`i mod 256 = 1`), then the window occupies the first 2 pixels in a row and `pixel_value_1` which is beyond the left margin is taken to be 0.Consequently,

`init_result <= pixel_3_value - 2 * pixel_2_value`

(c) If (`i mod 256 = 0`) then the window occupies the last 2 pixels in a row and `pixel_value_3` which is beyond the right margin is taken to be 0.Consequently,

`init_result <= pixel_1_value - 2 * pixel_2_value`

We now have to clamp `init_result` to `final_result` depending on the range of its value which simply follows the specification of the assignment.

- `memory_management`

  This is the most non-trivial process to model. The explanation of teh same is as follows:

  (a) Assignment of `rom_address` : **1 clock cycle**
  (b) Reading value from ROM : **1 clock cycle**
  (c) Arithmetic operations and variable assignment without directly accessing from memory : **1 clock cycle**
  (d) Assignment of `ram_address` : **1 clock cycle**
  (e) Writing value to RAM : **1 clock cycle**
  (f) Time taken for RAM to stabilise before data can be written or read: **3 clock cycles**.

  Consequently,we have introduced a counter `check` which for certain rising edges of the clock,is simply incremented which effectively serves as waiting for the corresponding operation for 1 clock cycle.`check` attains a total of 8 values before cycling back to 0.

  (a) For the case when pixel count `i` <= 65536,the process manages the above mentioned operations. In the intermediate trial when `check` = 4,the process manages the assignment of pixel values from corresponding vectors as well assigns `ram_address`.
  (b) When the pixel count `i` exceeds 65536,the value of `write_enable` is set to '0' which serves as a stimulus for a variety of processes concerning the VGA display.

- `ram_adjustment_for_display`

  (a) This is a crucial process which serves to update ram_address_while_disp which would update `ram_address` via another process.
  (b) The mathematical logic used for the same is as follows:

  ```
  ram_address_when_disp <= std_logic_vector(to_unsigned((256*(vpos)+ hpos),16));
  ```

  if the image is displayed in the upper right corner of the VGA display.

To better understand the following processes, it is necessary to understand the below details of a VGA display-

(a) HFP (Horizontal Front Porch)

HFP is a part of the horizontal blanking interval in a video signal. It is the period of time between the end of the active video (displayed image) and the start of the horizontal synchronization pulse. During the HFP, the electron beam in a CRT display or the timing of the display controller moves from the right edge of the screen to the left edge to prepare for the next line.

(b) VFP(Vertical Front Porch)

VFP is similar to HFP but pertains to the vertical blanking interval. It is the period between the end of the active display area and the start of the vertical synchronization pulse. During VFP, the electron beam or display controller moves from the bottom to the top of the screen to prepare for the next frame

(c) HSYNC (Horizonal Sync)

HSYNC is the horizontal synchronization pulse or signal. It is a short pulse that indicates the start of a new line on the display. HSYNC is used by the display to synchronize its scanning with the incoming video signal.

(d) VSYNC (Vertical Sync)

VSYNC is the vertical synchronization pulse. It is a short pulse that indicates the start of a new frame on the display. VSYNC is used by the display to synchronize its frame rate with the incoming video signal.

(e) HBP (Horizontal Back Porch)

HBP is the period of time between the end of the horizontal synchronization pulse and the start of the active video. During the HBP, the electron beam (or display controller) returns from the left edge of the screen to the starting position of the next active video line.

(f) VBP (Vertical Back Porch)

VBP is similar to HBP but pertains to the vertical blanking interval. It is the period of time between the end of the vertical synchronization pulse and the start of the active video for the next frame. During VBP, the electron beam (or display controller) returns from the top to the bottom of the screen to prepare for the next frame.

- `Synchronise_vertically`

(a) The primary purpose of this process is to generate the horizontal synchronization pulse (hsync). It does this by checking the value of hpos (the current horizontal position) against specific horizontal timing parameters, including HD (Horizontal Display), HFP (Horizontal Front Porch), and HSP (Horizontal Sync Pulse).

(b) If hpos is within the range (HD + HFP) to (HD + HFP + HSP), it sets hsync to '1', indicating the start of the horizontal synchronization pulse. If hpos is not in this range, it sets hsync to '0', indicating that the display is in the active display area. This logic ensures that the horizontal synchronization signal is generated accurately.

- `Synchronise_vertically`

  (a) Synchronization Logic: This process is responsible for generating the vertical synchronization pulse (vsync). It checks the value of vpos (the current vertical position) against vertical timing parameters, including VD (Vertical Display), VFP (Vertical Front Porch), and VSP (Vertical Sync Pulse)

  (b) If vpos is within the range (VD + VFP) to (VD + VFP + VSP), it sets vsync to '1', indicating the start of the vertical synchronization pulse. If vpos is not in this range, it sets vsync to '0', indicating that the display is in the active display area. This ensures proper vertical synchronization.

- `enable_video`

  (a) Video Display Logic: This process controls the videoOn signal, which determines whether the video display is enabled. It checks both hpos and vpos to determine if they are within the specified ranges (HD and VD) for the active display area.

  (b) If both hpos and vpos are within these ranges, it sets videoOn to '1', enabling the video display. If either hpos or vpos falls outside these ranges, it sets videoOn to '0', disabling the video display. This logic ensures that the video display is enabled only within the active display area.