UNIVERSITY OF CALGARY

ENEL 453

Laboratory #4 – Simple VGA Clock

Lab: B06

Kyle Derby MacInnis

Sajna Massey

03/08/2013

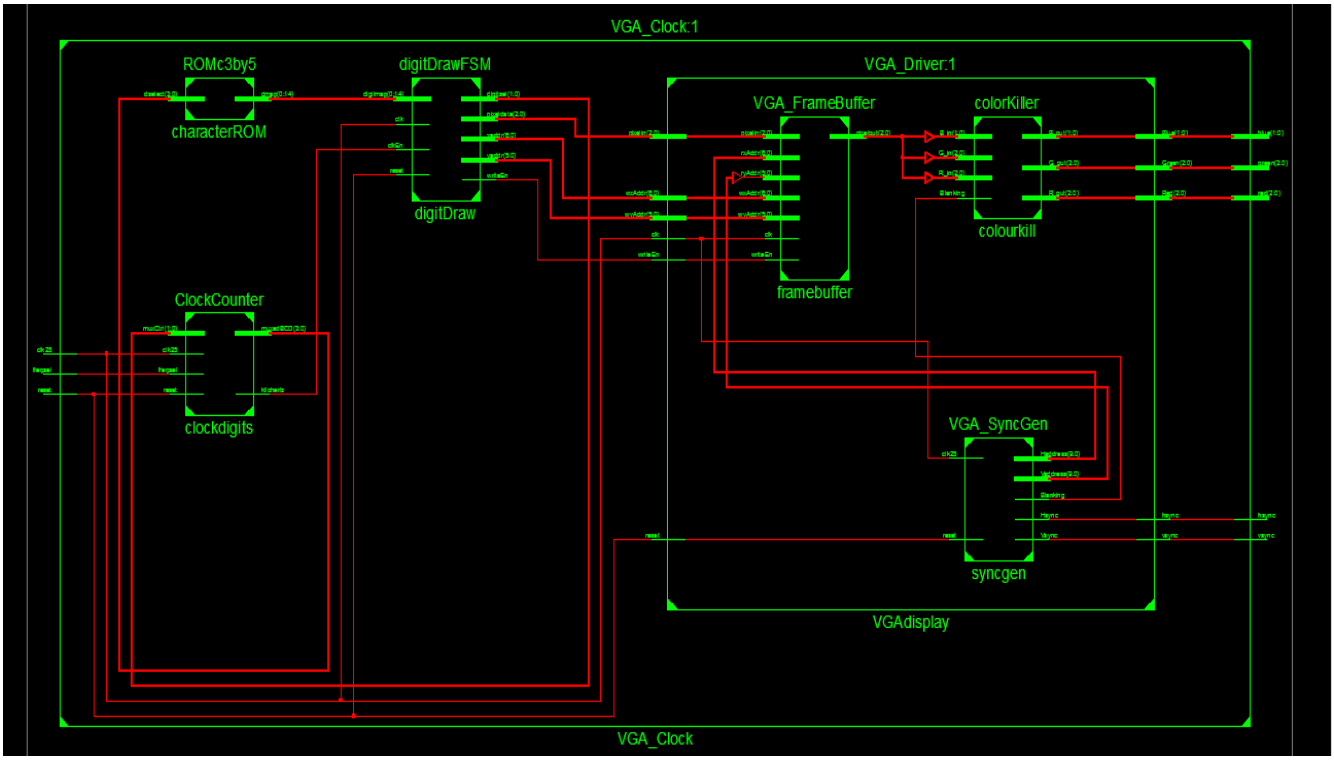
We declare that this laboratory report is entirely our own work and includes no material which has been copied from any other source excepting that material which is clearly identified as the work of others.

Abstract: The purpose of this laboratory is to design and implement a simple clock similar to the model designed in laboratory #2 with the distinction that it is displayed via a VGA connection to a monitor. This laboratory combines portions of both laboratory #2 and #3 and adds additional components as will be explored.

Design Theory: In order to properly implement a VGA clock, a few components are required: a clock counter, a digit FSM, a ROM based character map, RAM for read/write pixel data, a VGA buffer, and a VGA driver.

- The clock will be similar to the one used in lab #2, with the exception that it will not be outputting to a 7-segment display, but rather to the ROM block.
- The character maps for each digit will be stored in a predetermined ROM block based on their relative display locations. This outputs the character map to the digit FSM.
- The digit FSM will be in control over the mapping of the characters onto the current Frame. The characters are written into the dual-port RAM.
- The RAM will be dual-port, allowing for concurrent read/write operations, and it will be done synchronously. In addition, there will be a Write Enable signal for access control.
- The VGA buffer will be in control over the individual frames to be written onto the monitor and will read the pixel data out of the RAM block. This is contained within the VGA driver component.
- The VGA driver will function as the generator for the VGA signal and will contain substructures for Horizontal and Vertical Sync Generation as well as outputting the final pixel data.

Synthesis: After the initial design phase, the components were programmed using VHDL and synthesized via XST. The following RTL diagram of the final system was generated:



Simulation: After designing the functional description of the individual components it is imperative that they be tested prior to actual implementation. This is done via the use of test benches. The following figures showcase the waveforms corresponding to their respective components as labeled:

ROMc3by5: As can be seen, the ROM acts as intended and each input(BCD) results in the correct character map being generated, or if it is not a proper BCD code it results in an error being generated.

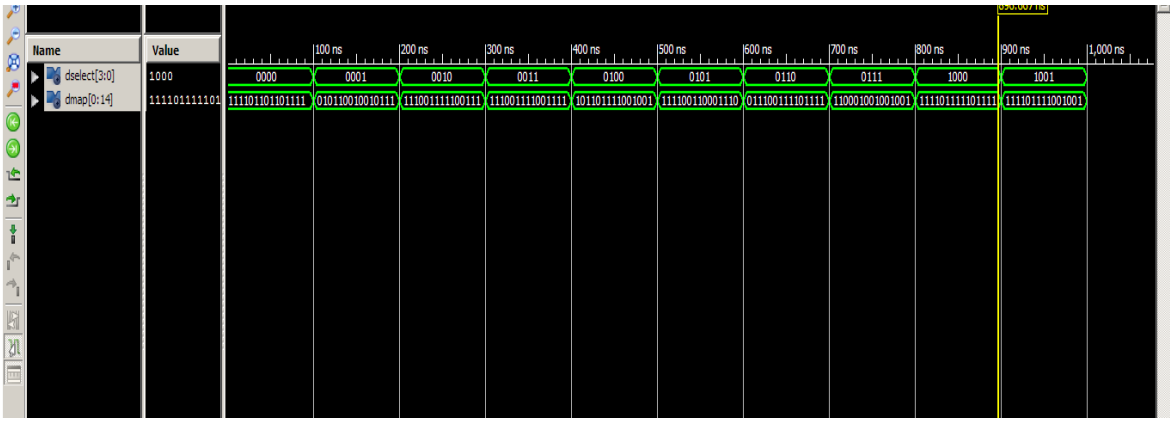


Illustration I: ROM Behavioral Simulation

VGA_FrameBuffer: Once again, it can be seen below that if writeEn is set, then data is written to the RAM block, whilst on every clock cycle it is being read from RAM. The new data is then available to be read as soon as it becomes written provided the address for reading is correct.

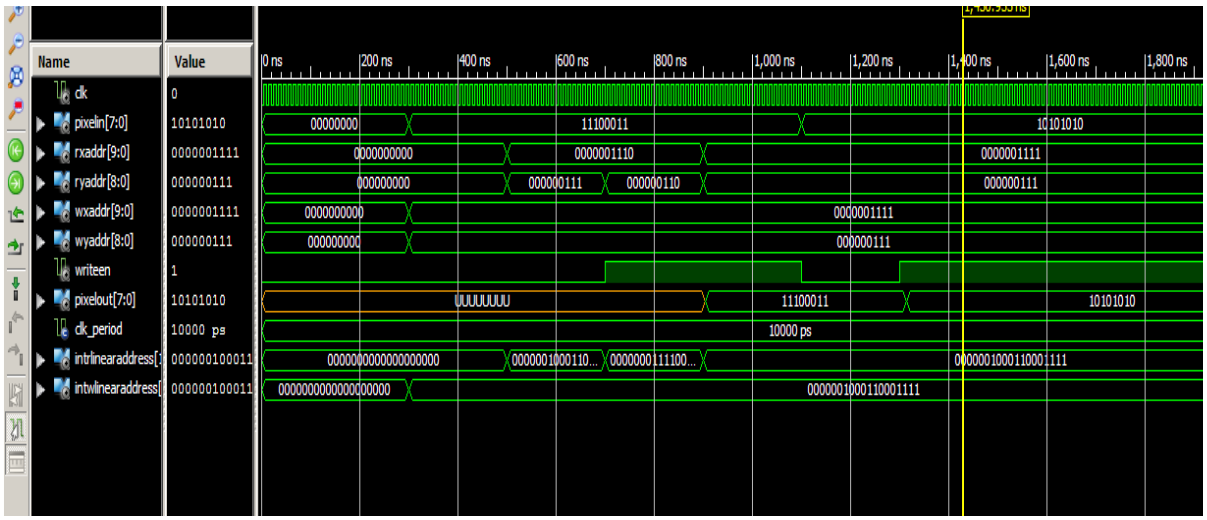


Illustration II: VGA Framebuffer Behavioral Simulation

VGA_Driver: The waveform for the simulation for the VGA Driver is shown below. The issue with this simulation is that the VGA sync generator is internal and drives the read address from the frame buffer while the write address is controlled externally. Since the write address is not being properly driven it is constantly unsure what the pixelout data is as it hasn't been written. What can be seen from the diagram is that at the marker shown, the read address is currently reading an address which has been previously written and should it be implemented with the correct write operations being done, no problems would occur. The VGA driver is working correctly in this case.

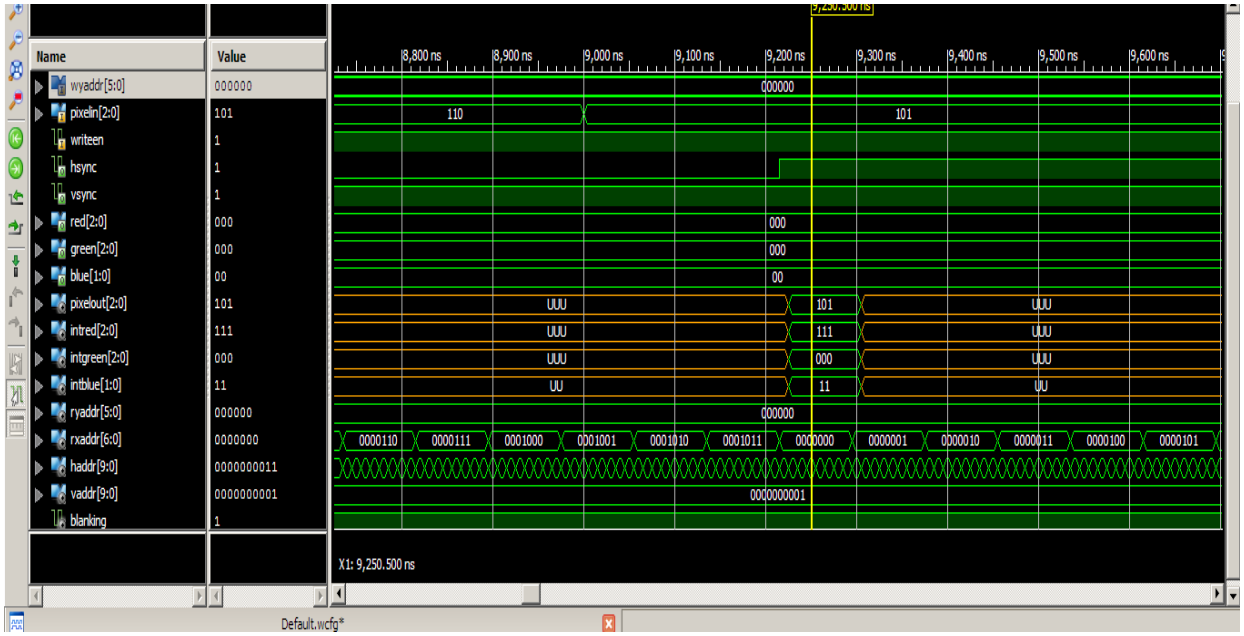


Illustration III: VGA Driver Behavior Simulated Waveform

Post Translate Simulation: The post translate simulated waveform can be seen below. As can be seen, the figure accurately describes the first frame of execution to the monitor. The first marker(yellow) outlines the beginning of the vertical sync signal which begins one frame. The second marker (middle blue) shows a portion of the waveform which outputs the characters to the screen(in this case just the first minute character). Since it is zoomed out quite far the individual pixels cannot be seen to change, only that there is change during this portion. The last marker(blue right) shows the transition from the first minute state(“11”) to the second minute state(“10”) as well as the characters positional map addresses are shown below on the bottom two lines of the waveform.

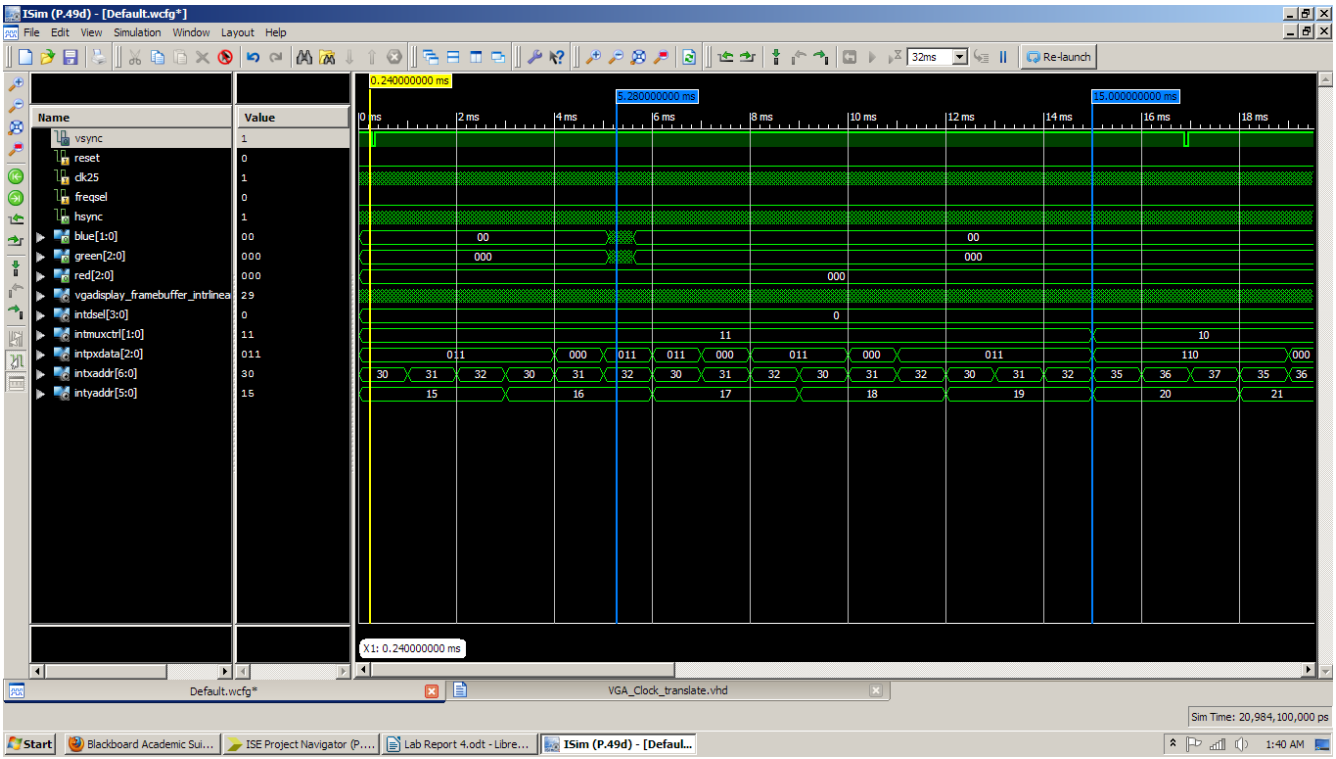


Illustration IV: VGA Clock Post Translate Simulated Waveform

Additional Questions:

1. Dual port RAM allows for both reading and writing to occur at the same time as there is a separate bus for reading data and a separate bus for writing data. The writing is controlled through the use of a control signal: write Enable. Single port RAM only allows for either reading or writing to occur at any given time. It is done asynchronously and there is only one bus for data transfer. So if reading is being done, data cannot be written and vice-versa.
2. From the perspective of interface signals, the differences between ROM and RAM are mainly due to the nature which data is accessed. With ROM it is a read only action, and correspondingly this results in a value being sent in via an input bus, and an output coming from the output bus. This is not a synchronous action and as such there is no clock input as it is essentially instantaneous. With RAM there is both a capacity for reading and writing and as such there is data being saved and read from memory. RAM in this case is also synchronous and as such it has a clock input and a writeEn signal to dictate when to read, and when to write which contrasts the nature of ROM.
3. The role of the Frame buffer in this project is to allow for the display frame to be written and read from memory (RAM). It acts as the main container for the pixel mapped monitor frame.
4. The clock would have to be a 40MHz clock or higher to allow for proper synchronous VGA output to be allowed, but barring that, all one would have to

do to adjust for an 800x600 display would be to change the VGA SyncGenerator to output the proper timing. Since the actual position and size of the clock are not dependent on the resolution(barring too small of one) it would merely scale the apparent size of the clock against the screen. Now, if one were to account for the scaling factor and the position then it would require that the position of the digits be changed in the digitDrawFSM component, as well as the down sample value would need to be adjusted. Finally, the character maps would need to be scaled up accordingly and this would require more ROM space.

5. In order to allow for more character maps to be stored in ROM, and for them to be properly displayed a few changes would need to be implemented. The first change would be for any addition character maps to be created which is merely designing them on 3x5 matrix and then converting them to the corresponding map. Secondly, depending on how many characters the ROM would require a potential larger bus for input (26 letters+10 numbers = 36 maps => 6 bits). Finally the last step would be to implement a new digitDrawFSM component in order to properly map out the pixels to be drawn if the text was to be state based otherwise it could just be fed into the RAM block if it was at constant positions.
6. The process for implementing a circuit to draw a vertical or horizontal line between two arbitrary points would be similar in design to the digitDrawFSM component. The code would consist of input selecting the two points and whether it is horizontal or vertical, and a simple conditional check for whether or not the current position count(address on screen) was within this range. In addition it would be prudent to check to see if the points were in actual fact along the same vertical or horizontal line as well as to determine which colour was to be used for the line.