Design & Implementation of VGA Controller on FPGA

Dr Sagar Patel[1] & Shalini Mishra[2]
[1]Assistant Professor,[2]Student
[1]Electronics & Communication, [1]Charusat University, Anand,India

Received: January 28, 2019 Accepted: March 03, 2019

ABSTRACT: *As a standard display interface, VGA (Video Graphics Array) has been widely used. This paper represents the design and implementation of VGA controller. The paper has given its top layer module design. Detailed information is focused on the system architecture and software programming. This controller is developed using only Verilog HDL (hardware description language). The system can display various English alphabets and digits.*

Key Words:

1. INTRODUCTION

VGA (video graphics array) is a video display standard. It provides a simple method to connect a system with a monitor for showing information or images. As a standard display interface, VGA has been widely used. There is more and more
need in displaying the result of the process in real time as the fast development of embedded system, especially the development of high-speed image processing
In display of a screen, the scanning process starts from row 0, column 0 in the top left corner of the screen and moves to the right until it reaches the last column. When the scan reaches the endof a row, it retraces to the beginning of the next row. When it reaches the last pixel in the bottom right corner of the screen, it retraces back to the top-left corner and repeats the scanning process, refer to (Fig. 1). In order to reduce flicker on the screen, the entire screen must be scanned 60 times per second. This period is called the refresh rate. The human eye can detect flicker at refresh rates less than 30 Hz.
Within a CRT display, current waveforms pass through the coils to produce magnetic fields that deflect electron beams to transverse the display surface in a *raster* pattern, horizontally from left to right and vertically from top to bottom. As shown in (Fig. 2).Information is only displayed when the beam is moving in the *forward* direction—left to right and top to bottom—and not during the time the beam returns back to the left or top edge of the display. Much of the potential display time is therefore lost in *blanking* periods when the beam is reset and stabilized to begin a new horizontal or vertical display pass.
The monitor screen for a standard VGA format contains 640 columns by 480 rows of picture elements called pixel. The fundamental parameters of a screen are clearly mentioned in (Fig. 3). Based on the mentioned resolution, using 25Mhz pixel clock and 60 Hz refresh rate, the timing signals are pre-defined and hence the pixel and lines are deduced, for further details kindly refer the attached appendix.
VGA control timing is depicted in (Fig. 4). The figure shows the relation between each of the timing symbols. The timing for the sync pulse width (TPW) and front and back porch intervals (TFP and TBP) are based on observations from various VGA displays. The front and back porch intervals are the pre- and post-sync pulse times. Information cannot be displayed during these times.

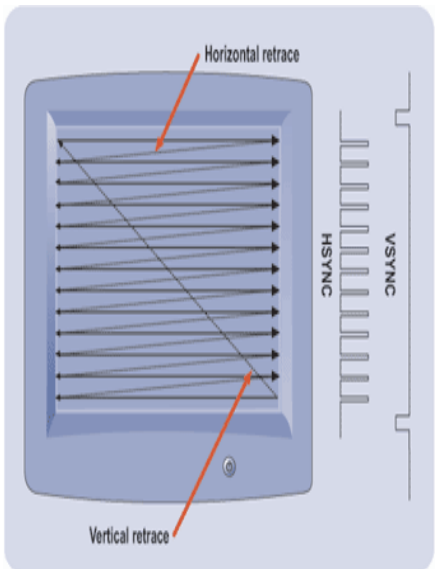

Fig. 1

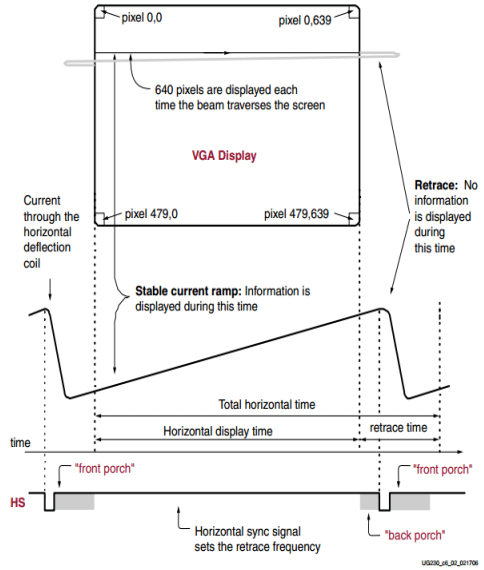

Fig. 2 [1]

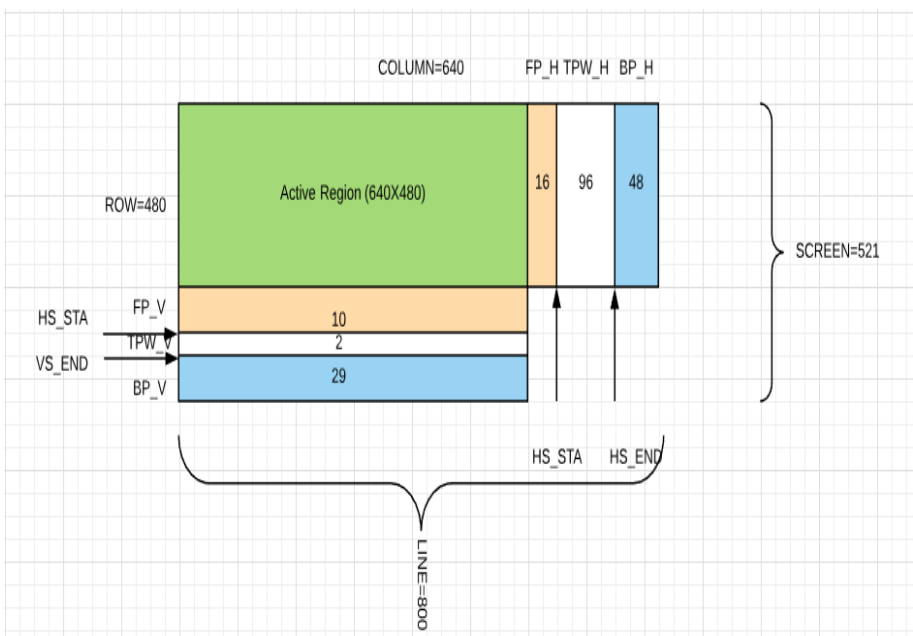

Fig. 3

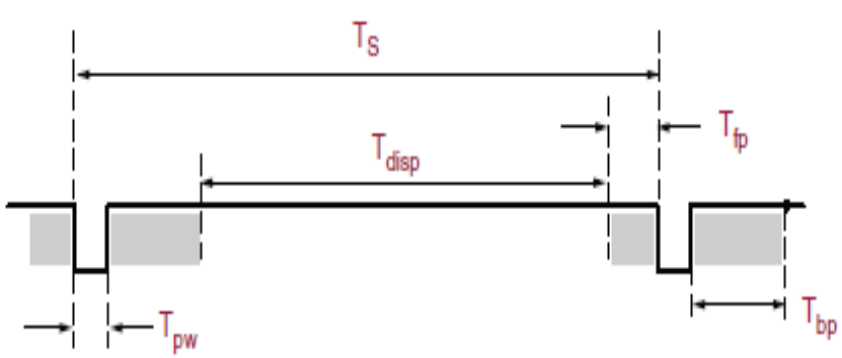

Fig. 4 [2]

2. Literature review

The project is broadly divided into smaller blocks keeping the standard timings in mind for 640X480 resolution and refresh rate of 60 Hz. Basically, videos or movies are series of still images. If they may swiftly, at the rate of 50-60Hz wherein the movement is not visible. Thus, moving images result in movies or videos. Each block is briefly explained here.

Block – 1

This block is driven by user, based on user's wish to "enter" or "delete" a character the desired input is stored in a RAM of size 6X6, which shall later on drive rest of the blocks. This will result in display of 2 rows with 3 characters each (this can be easily altered for a number of characters).

Block 2 –

(Fig. 5) demonstrates the FSM of the logic, wherein the mentioned states are – IDLE, TRANSFER, TRIGGER.
IDLE state determines the number of characters in row and number of rows to be displayed.
TRANSFER state links LUT with RAM for the selection purpose in accordance with the user.
TRIGGER depicts the selection of desired LUT, it's enlargement and transfers the same according to the specifications to an array of size (16X71) with the desired gap.

Block – 3

(Fig 6) is responsible for index generation which changes with the refreshing of the array

Block – 4

(Fig 7) depicts the FSM for active and blanking region and gives the sequential output accordingly.

Holistic –

(Fig. 7) clearly depicts the block diagram of the entire program. RAM temporarily stores the input provided by the user with number of rows as the number of inputs by the user and one column. Sync pulse generator with the help of i_clk generates h_sync and v_sync the separate 2 lines and frames. From an LUT the data is enlarged and thenplaced in the array in accordance with the required gap and user given inputs. This is then refreshed and given sequentially as an output.

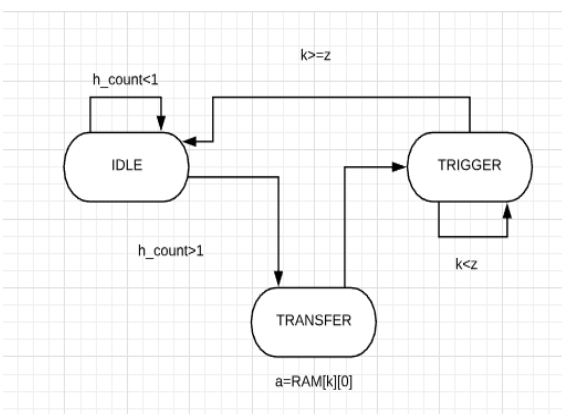

Fig. 5 [5]

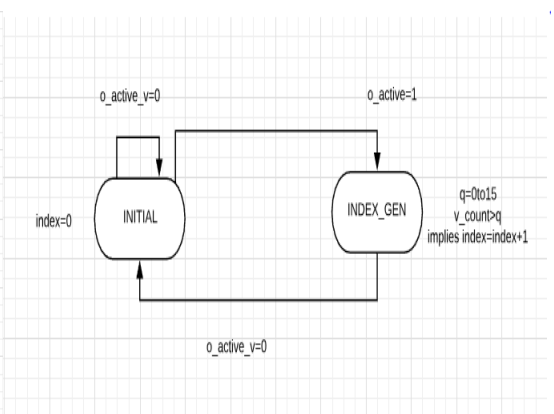

Fig. 6 [5]

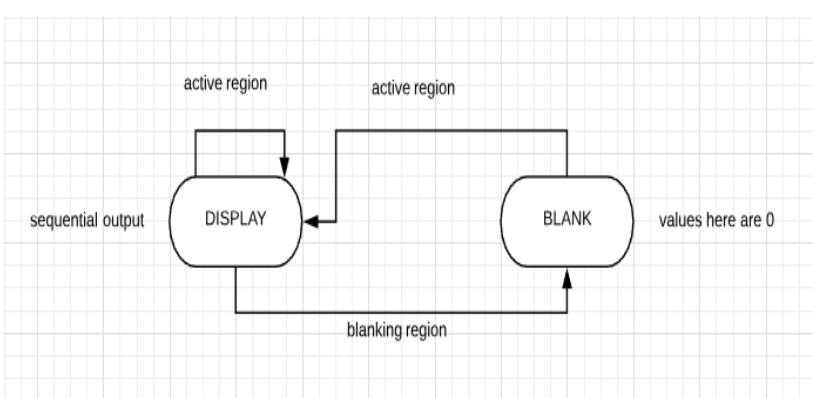

Fig. 7 [5]

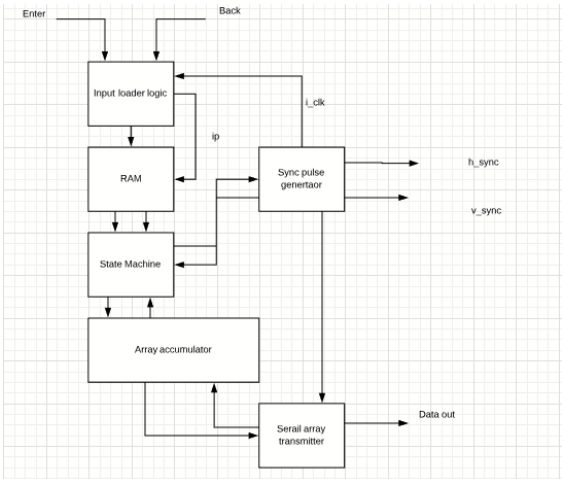

Fig. 8 [5]

3. Conclusion

Summing it up, FPGA (Field Programmable Gate Array) is indeed helpful invention in order to develop a VGA controller. Through Verilog HDL (Hardware Description Language) on FPGA VGA could be constructed easily without constructing the circuit manually, just to write a behavioral model or few behavioral models based on its logic flows, then simulate it with test benches, synthesize it with netlist, and finally program it onto FPGA. It is very effective as this VGA Controller only needs new data to change to other design display. Thus, FPGA-based VGA controller might be a good choice as it is easy to be designed and tiny to be used.

4. References

1. https://www.researchgate.net/publication/288835032_FPGA_Implementation_of_CORDIC_Processor
2. https://www.cnblogs.com/shangdawei/p/4760933.html
3. https://www.manualslib.com/manual/1248711/Xilinx-Spartan-3.html?page=24#manual
4. https://www.manualslib.com/manual/1248711/Xilinx-Spartan-3.html?page=24#manual
5. https://www.lucidchart.com/documents/edit/6858d9ba-6909-4bda-86b3-2786cbc066cd/0

5. APPENDIX [5]

Horizontal Timing –

Pixel clock = 25 MHz Pixel time = 0.04 μs
Horizontal video = 640 pixels x 0.04 μs = 25.60 μs
Back porch, BP = 48 pixels x 0.04 μs = 1.92 μs
Front porch, FP = 16 pixels x 0.04 μs = 0.64 μs
Sync pulse, SP = 96 pixels x 0.04 μs = 3.84 μs
Horizontal Scan Lines = SP + BP + HV + FP= 96 + 48 + 640 + 16 = 800 pixels x 0.04 μs = 32 μs
1/60 Hz = 16.67 ms / 32 μs = 521 horizontal scan lines per frame

Vertical Timing –

Pixel clock = 25 MHz Horizontal scan time = 32 μs
Vertical video = 480 lines x 32 μs = 15.360 ms
Back porch, BP = 29 lines x 32 μs = 0.928 ms
Front porch, FP = 10 lines x 32 μs = 0.320 ms
Sync pulse, SP = 2 lines x 32 μs = 0.064 ms
Vertical Scan Lines = SP + BP + VV + FP = 2 + 29 + 480 + 10 = 521 lines x 32 μs = 16.672 ms
1/60 Hz = 16.67 ms