Implementing VGA Application on FPGA using an Innovative Algorithm with the help of NIOS-II

Ashish B. Pasaya[1]
[1]E & C Engg. Department, Sardar Vallabhbhai Patel institute of technology, Vasad, Gujarat, India.
Kiritkumar R. Bhatt[2]
[2]Associate Professor, E & C Engg. Dept.,
Sardar Vallabhbhai Patel Institute of Technology, Vasad-388306.

Abstract
Basically, here we have used VGA for implementing basic graphics applications that can be either used in a single user game or either in advertisements that deals with real-time application. Further expanding the logic with the coding part even double user game could be developed. So, we thought of using VGA as a standard for this implementation as it is the basic graphics array and compatible with other graphical arrays. Here, we used HDL language on Quartus-II software for interfacing the required peripheral to the NIOS-II Soft-core processor through FPGA Cyclone-II Processor. Where, we made use of the Innovative Algorithm for implementing the application of VGA with the help of C language on NIOS-II Soft-core processor that will contain the logic part. Finally, the results that we obtained for VGA application implementation.

Keywords: VGA, FPGA, HDL, Quartus-II, NIOS-II, DE2 Education Board.

I. Introduction

In this project the most required things will be VGA, DE2 Education Board on which the FPGA and NIOS-II soft-core processor is embedded along with other components. Here, VGA stands for "Video Graphics Array" (en.wikipedia.org/wiki/Video_Graphics_Array). It is the standard monitor or display interface used in most PCs. Therefore, if a monitor is VGA-compatible, it should work with most new computers. The VGA standard was originally developed by IBM in 1987 and allowed for a display resolution of 640x480 pixels. The VGA supports both All Points Addressable graphics modes, and alphanumeric text modes. There are two kinds of VGA interface signals to display. One is data signal, and the other one is control signal. There are three data signals red, green and blue and two control signals horizontal synchronization and vertical synchronization signals [1]. There are different frequencies of the horizontal synchronization signal and vertical synchronization signal for the changeable output resolution [1]. Thus, if someone wants to implement any application on any higher graphics arrays, the try could be given on the VGA first. So, we did the same thing by implementing the application on the VGA.

Now, if we talk of FPGA then we used Cyclone-II FPGA of family EP2C35F672C6. For this we used Quartus-II software in which you can specify your design by three ways i.e. schematic entry, Verilog HDL and VHDL. In our case we have used the Verilog HDL to specify the design. It is possible to make use of even both the HDL languages together. Also, it is possible to implement the required application by just using the HDL languages but then this will make your code very lengthy and will also increase the complexity. Due to which it will reduce the easiness to understand the code and it will also put burden of the memory associated with the FPGA. Now, we know that the memories associated with the FPGA will have sufficient memory space for small information's like limited number of characters and some image requiring very small memory to be displayed on screen, but would be insufficient to display an animated characters or images. For this kind of situation we will require some additional processor that can reduce the length of code and other information while developing some real-time applications. So, we do have a processor named NIOS-II soft-core processor which allows the input in both the C and C++ languages. This processor is embedded on the DE2 Board itself.

Thus, what we need to do is just to interface the peripheral required for our application with the NIOS-II soft-core processor. For that here we use the HDL language to interface the NIOS-II with required peripheral through FPGA. We had used C as an input language on the NIOS-II soft-core processor for implementing the innovative algorithm that we developed for our required application. Here, our required application is a single user game. Where there is a mouse as an input for single user and VGA screen as an output. On screen we will have one paddle which in our case is a rectangular block and one small ball like structure which in our case is small square block. Paddle will move according to the movement of the mouse that is connected to the USB blaster of the DE2 Board and the ball will do the movement depending on whether there has been a click on a mouse or not.

This paper is organized as follows: Section 2 describes the innovative algorithm used for implementing VGA application. Section 3 shows the VGA application implementation results. Section 4 concludes the paper.

II. Innovative Algorithm for VGA application

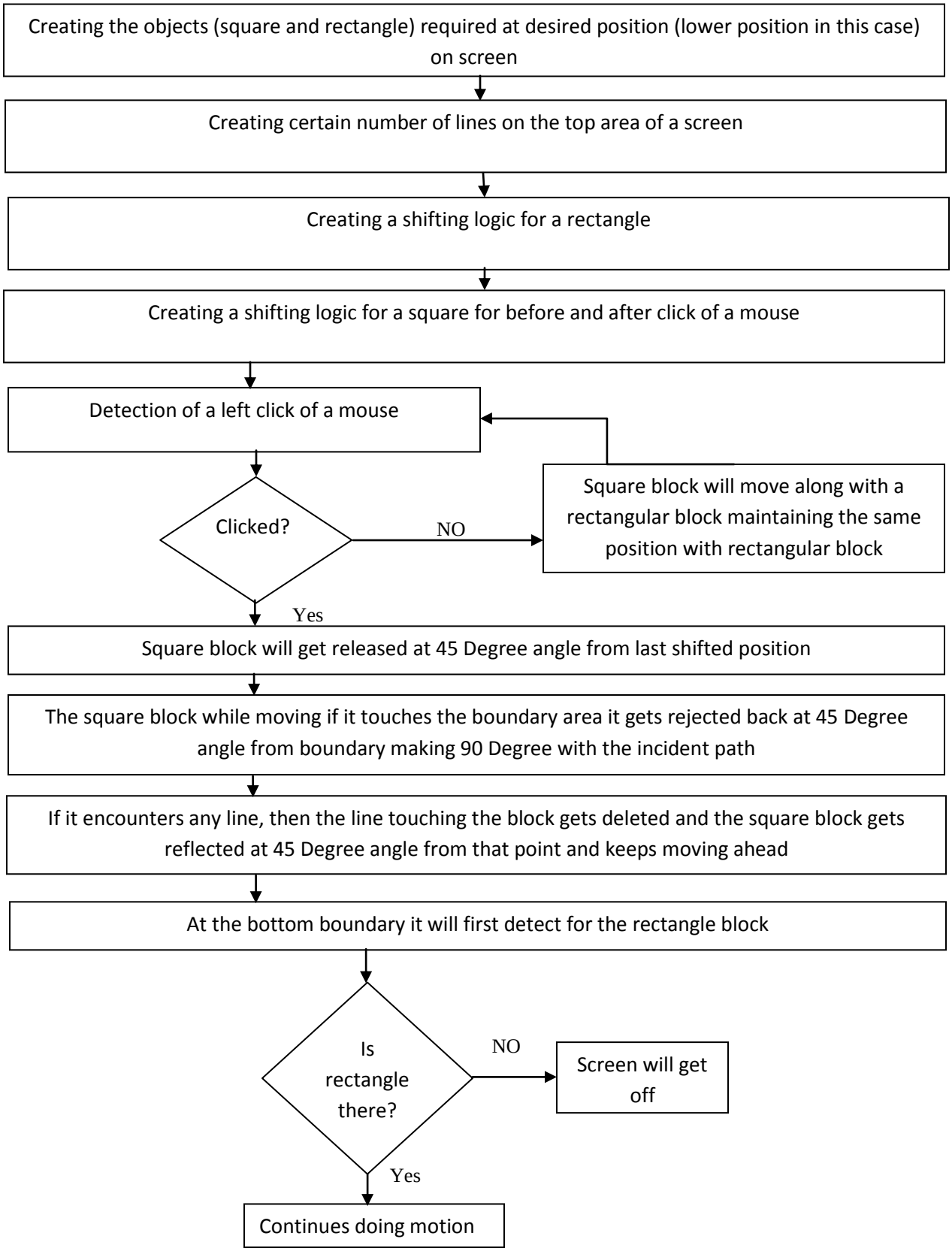

Figure 1. Innovative Algorithm for VGA Application

Here, first of all we will create the objects that are required in our project to be displayed on the screen i.e. a small square block, a rectangular block and few lines. Where we will have these lines at the top most part of the screen and these rectangle block will do movement in horizontal plane on the bottom part of the screen and will remain at the fixed at vertical position of the screen, while the small square in this project will have both the horizontal movement as well as other random movement depending on the situation.

Now, we will create a shifting logic in which the rectangle which shift in horizontal direction on screen being fixed in vertical direction. This shift will be according the movement of the mouse, where the movement of the mouse decides whether to get shifted in on left or right. That is in wherever direction the mouse moves the rectangle block on the screen also moves, just because the rectangle block is in synchronization with the position of the mouse. In other sense we can say that the rectangular block represents the cursor of the mouse but not exactly. The movement of the rectangle block in the screen would be only within the boundaries set by the programmer on the screen. For example, you can take the boundary as 620×460.

Now creating a shifting logic for a small square block. This task will be more complex than that of creating a sifting logic for a rectangular block. Here there will be two kind of motion of a square block depending on the click of the mouse. Here the value will be assigned to one variable which will detect whether the click has occurred or not. Like its value will be set if the click has already occurred and unset if not occurred. This part will be the detection part of whether the click has occurred or not. Now during detection if no click has occurred then it will just keep shifting in horizontal direction along with the rectangular block, where the position of a square block will remain constant with respect to the position of the rectangular block. These detection will occur continuously. If the click occurs during detection then the square block will now get released at 45 degree angle from the last moved position and will start to get incident on the boundary wall set by the programmer. Now this angle of incident and angle of reflection at the wall will make 90 degrees with each other. After getting reflected the square block will keep moving ahead and will touch the bottommost line at the top of the screen. When the square block touches the line it gets reflected from the like at 45 degrees again making 90 degree angle with the incident path to the line and the line will get deleted/erased from that position. The square block will keep moving ahead and now will touch the another boundary wall and get reflected back at 45 degree angle making 90 degree with the incident path.

Now it keeps doing motion ahead in the bottom direction where the rectangle block is already doing motion as per the movement of the mouse in horizontal direction. The rectangular block must catch the square block on its top otherwise if it misses the square block then the screen will get off. There will be detection by square block logic for the presence of the rectangular block on the bottom surface of the screen. If the rectangular block catches the square block on the top then the square block will get reflected from the top surface of the rectangular block at 45 degree angle making 90 degree angle with the incident path and keeps moving ahead towards wall and will follow the same scenario as ahead it did and will erase each lines when it gets touched to line and will then incident to the wall after each line gets erased by the touch. Here when the square block is shifting then the one pixel from back position will get and one pixel in front of the rectangular block will be displayed which shows that the rectangular block is doing motion if speed is increased for the shift.

The rectangular block will move according to positive X and negative X direction. While the square block move according to both the positive and negative direction of X and Y. so, at whatever wall it gets incident to the wall or any other object it will get reflected in its negative direction.

Here the square block will become independent of the movement of the mouse when the click occurs, so there will be no dependence of the square block on the rectangular block and the mouse as well. The speed of the movement of the rectangular block and square block can be set between the minimum 50 MHz to maximum 400 MHz in our project.

III. Results of VGA application implementation

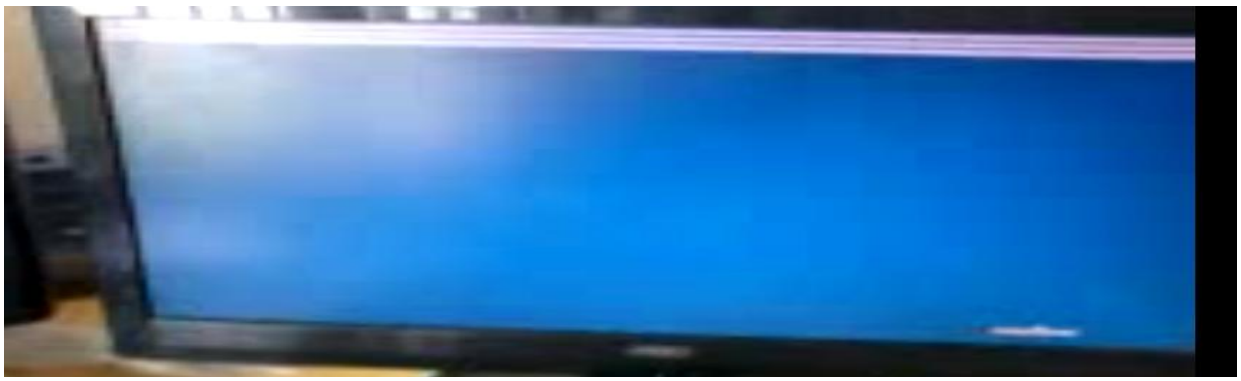

FIGURE 2. Motion of a square and a rectangular block

FIGURE 3. Release of a square block at 45 degree angle

FIGURE 4. Motion of a square block towards the lines after getting reflected from the side wall of a screen

FIGURE 5. Erasing of a line with which the square block got incident

IV. Conclusion

In this paper we presented an innovative algorithm for implementing basic application of VGA for a single user game and the output that we attained. It further can be extended into an advertisement with real time application or a 1double user game with respective addition of logic through NIOS-II processor. We interfaced NIOS-II with required peripheral through FPGA using HDL languages and used C as a language for implementing the innovative algorithm. So here NIOS-II adds more flexibility along their use with FPGA and reduces the complexity in coding part especially, which makes it suitable to meet both our hardware and software real-time requirements.

Acknowledgement

Words are often less to revels one's deep regards. An understanding of work like this is never the outcome of the efforts of a single person. First off all i would like to thank the Supreme Power, one who has always guided me to work on the right path of the life. Without his grace this would never come to be today's realty. This work not have been possible without the encouragement and able guidance of my supervisor Prof. K.R.Bhatt, his enthusiasm and optimism made this experience both rewarding and enjoyable. Also my sincere thanks to Mihir Sir and Nirav Sir (smartech solutions), Vadodara for providing me tools and guidance required in my work.

References

[1] Guohui Wang and Yong Guan, "Designing of VGA Character String Display Module Base on FPGA", IEEE 2009.
[2] Cyclone II Device Handbook (PDF) – Altera.
[3] DE2 Development and Education Board User Manual.
[4] NIOS II Software Developer's Handbook (PDF) – Altera.
[5] Fawcett, B., "FPGAs as Reconfigurable Processing Elements", Circuits and Device Magazine, March 1996, pp. 8–10.
[6] Van-Huan Tran and Xuan-Tu Tran, "An Efficient Architecture Design for VGA Monitor Controller", IEEE 2011.
[7] Babu T Chacko and Siddharth Shelly "Real-Time Video Filtering and Overlay Character Generation on FPGA", IEEE 2011.
[8] A. D. Ioan, "New Techniques for Implementation of Hardware algorithms inside FPGA Circuits", Advance in Electrical and Computer Engineering, Vol. 10, No. 2, Suceava, Romania, 2010.