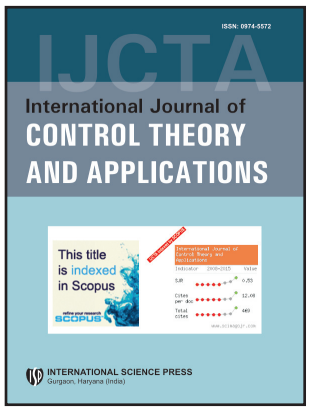


**International Journal of Control Theory and Applications**

ISSN : 0974-5572



**Volume 9 • Number 44 • 2016**

# Design and FPGA Implementation of Space Shoot Game

**Akarsha Mishra[a], Adesh Kumar[b] and Rakshita Parihar[c]**

*[a,c]M.Tech Scholar, Embedded Systems with specialization in wearable technology, University of Petroleum & Energy Studies, Dehradun, India. Email: [a]Akarsharr1@gmail.com; [c]Rsparihar10@gmail.com*
*[b]Department of Electronics, Instrumentation and Control Engineering, University of Petroleum & Energy Studies. Dehradun, India. Email: adeshkumar@ddn.upes.ac.in*

***Abstract:*** The research article presents the design and implementation of an arcade style space-shoot video game using the retro cell phone game 'Space Impact' as inspiration. The game contains a spaceship and aliens. Spaceship has to shoot aliens using the missile and collect scores. The game is designed with hardware accelerated graphics output through VGA. The player controls the game with the three buttons of a ps/2 mouse. The project resulted in a functional game that integrates both hardware and software in order to display graphics and handle game logic. The game objective is to eliminate aliens that appear on the screen. There is no such death condition for the spaceship that if it gets hit by the aliens then it is dead. The game is over when all aliens are shot. The game is designed with the help of VHDL programming on Xilinx ISE 14.7 software. The synthesis of the game is done with the help of Spartan 3E FPGA which estimate the supporting frame 50 MHz.

***Keywords:*** Field Programmable Gate Array, Very high speed Hardware Description Language, Video Graphics Array, Personal System/2.

## 1. INTRODUCTION

The creation of games is always in the mind of the engineers. The scientists and engineers have always thought of creating new things for entertainment. The beginning of the electronics gave big opportunities to the engineers to create games.

Game system is divided [3, 4], division provides advantages. Finding solutions for minor problems is easier than for major ones. Moreover, FPGA usage is a great improvement for digital designing, it has [8] modular capability and VHDL programming is concise and clear. The game started with the idea of developing visual representations [1, 5] to convey data to the player like graphics of aliens and spaceship. The graphics are controlled by the VGA module [10, 12] that controls the screen size of the game and the control of movement of the horizontal and vertical porch of the display. Storage of graphics is done in BRAM and the graphics are processed by GPU [9, 20]. A Graphics Processing Unit (GPU) is a dedicated circuit that operates on a storage area (frame buffer) for the purpose of providing display output. The next step is the game control [6, 7], how

the graphics will move, shooting of the missile and dying of the targets[17]. Pixel plotting plays a great role in the game control since to shoot the targets, the pixels of the spaceship and the target had to be compared. Then comes how to move the spaceship with. Well that could have been done in various ways but this paper is based on the PS/2 [14,10] controlled movement. PS/2 sends and receives data serially through ps2d data line. [21] Porting of the PS/2 is done on the two pins of the FPGA board G13 and G14 that are mainly the data and the clock lines respectively. The mouse employs a relative coordinate system. The movement is registered in two dimensions: horizontal (X axis) and vertical (Y axis). Instead of the absolute coordinates, mouse sends the offset values when moved. These values are relative to the position the mouse is in prior to sending the data packet. Offsets are 9-bit wide and can be positive (up or right movement) or negative (down or left movement). The standard PS/2 mouse sends data using three consecutive packets. Each packet begins with the Start bit (always Low), followed by a data byte (starting with least significant bit), the Odd Parity bit and the Stop bit (always High). Data bytes contain mouse movement and button status information. Byte 2 and Byte 3 contain the motion values stored in two's complement format. Byte 1 contains the overflow bit, the sign bit and button status bit (M for middle, R for right and L for left). The main body of the game is composed of three parts: controlling inputs, processing logic, and displaying output. Controlling input part includes PS/2 mouse interface. Player use mouse to control the movement of the spaceship and shooting of the aliens. Logic processing part is using an FPGA Spartan 3E board to complete. The program is totally realized on the FPGA chip [18], and communicates with the controlling input and displaying output part through the corresponding interface. Display part transmits the VGA signals to the external monitor which shows the game display. Games depend not only on sophistication to be amusing but also on playability and well-done ideas. Second, it does not matter how good you could have planned a game, it can always be improved.

## 2. RELATED WORK

Many researchers have worked on the design and FPGA implementation of games. B. Bowman, N. Elmqvist, T.J. Jankun-Kelly, et. al., [1] have researched about employing statistical data graphics to present more data to the player. C. Xi [2] has discussed about the interfacing of PS/2 controller and in what fields it can be used. G. Wang, T. N. Tran, H. A. Andrade, et. al., [3] has provided graphical programming for different design patterns in FPGA. H. F. Jimenez, R. F. M. Gonzalez, P. V. G. Hernandez, A. D. Cedillo, et. al., [4] designed an fpga game where led's are controlled by the movement of the PS/2 mouse. J. Qu, Y. Wei, Y. Song, et. al., [5] the author focussed on FPS game development such as expressive play, game and character states management, rending scene, maps in game world, online and team communication, weapon management in game, and multi-enemy rendering. K. Liu, Y. Yang, Y. Zhu, et. al., [6] designed a tetris game where the players had to move or rotate the blocks with the ps/2 keyboard. M. Boule, Z. Zilic, et. al., [7] have researched about the growing advantages of FPGA over ASIC and show how some of the FPGA architectural features can be used to improve the design of the chess move generator. M. M. Zavlanos, G. J. Pappas, et. al., [8] have proposed a distributed approach for multiple player games. N. Zhar, M. A. Ali, M. Eleuldj, A. Raji, et. al., [9] has explained an fpga based image processing method. ] P. K. Gaikwad, et. al., [10] has discussed about the top module that he has developed for ps/2 interfacing with monitor. P. Lankoski, S. Bjork, et. al., [11] has researched on realizing characters for games inherited from human characters in the movies or novels to support interactive experience for the players. P. P. Chu, et. al., [12] have detailed about the working of ps/2 mouse as a unidirectional as well as bidirectional transmitter and have explained it through the code. R. Drechsler, N. Drechsler, et. al., [13] have presented a hardware description of evolutionary algorithms with the help of a tool GAME-HDL implemented in vhdl. R. J. Teather, J. Carette, M. Thevathasan, et. al., [14] have compared the performance of the player if the scaling i.e the display size varies. R. J. Teather, M. Thevathasan, J. Carette, et. al., [15] have also compared the performance based on two types of display scaling i.e uniform and non-uniform. ] R. P. McMahan, E. D. Ragan, A. Leal,

R. J. Beaton, D. A. Bowman, et. al., [16] has researched on the use of commercial video games as a means of enhanced learning, developing skills etc. R. Szabó, A. Gontean, et. al., [17] have developed a pong game with th control being done with the four fpga push buttons. S. Gao, S. N. Givigi, A. JG Beaulieu, et. al., [18] The author addresses the implementation of multi pursuit evasion games in which there are multiple agents doing their individual roles and interacting with the real world simultaneously. In such systems real time processing is an important issue which cannot be achieved by microcontrollers and therefore, FPGAs are a good solution for resolving such problems. An efficient learning algorithm is used for each agent to take a desired action at each step. ] S. N. Givigi Jr, H. M. Schwartz, et. al., [19] have shown how a pursuit-evasion game may be modeled with Markov chains. They have also shown that using a learning automata will lead to the optimal solution of the _nite time/_nite state game with incomplete information. S. Zafar, S. Kataria, A. Sharma, et. al., [20] explains how the graphics are processed in FPGA and where they are stored. The paper has very explained about the GPU (Graphics processing unit) that controls the display of graphics through VGA on screen. V. Alves, I. Cardim, H. Vital, P. Sampaio, A. Damasceno, P. Borba, et. al., [21] have discussed porting as a critical task in mobile game development which is much easier in desktop games.

## 3. STRUCTURE OF THE GAME

The block diagram of Space shoot game is shown in Figure 1 and is composed of 7 modules: mouse control module, game control module, text display module, and graphic display module, storage unit, multiplexing unit and VGA modules. The graphic display module and text display module share VGA control through multiplexing units; game control modules, ps/2 control module and memory module complete data exchange through the storage of data bit.

The shown block diagram defines the entire game structure. Main input of the game comes from the ps/2 mouse giving the status of left, right and middle button. This data in the form of signals is connected to the three signals nes_a, nes_left and nes_right of the graphics module. This module takes these signals as an indication to move the spaceship left or right or to shoot the targets. Each of the module is explained below in detail.

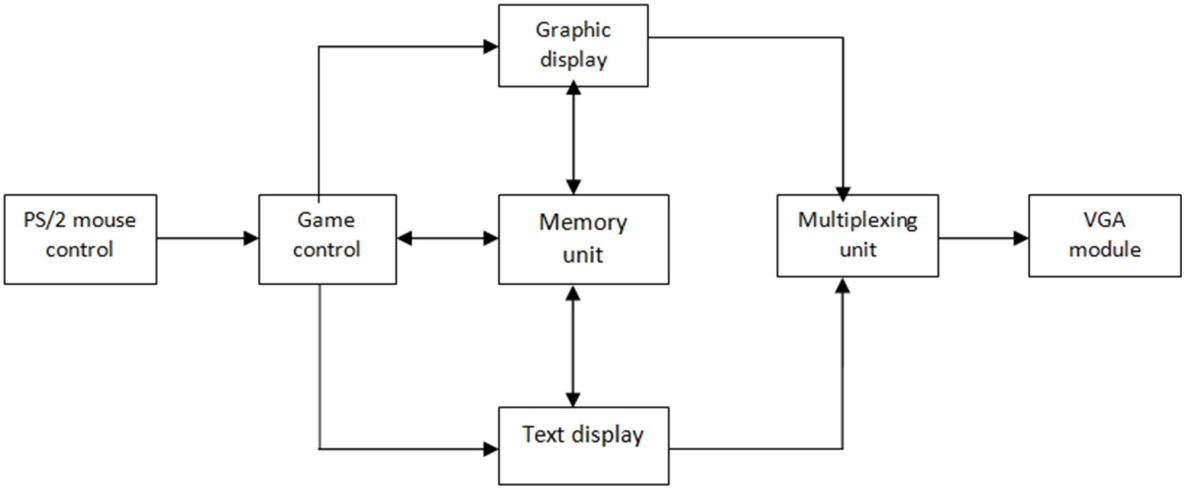


**Figure 1: System Block Diagram**

### A. Mouse Control

The position of the mouse pin can be understood with the help of Figure 2. A mouse generates a clock and data signal when moved; otherwise, these signals remain high indicating the idle state. Each time the mouse is moved,

the mouse sends three 11-bit words to the host. Each of the 11-bit words contains a '0' start bit, followed by 8 data bit (LSB first), followed by an odd parity bit, and terminated with a '1' stop bit. Each data transmission contains 33 total bit, where bit 0, 11, and 22 are '0' start bit, and bit 10, 21 and 32 are '1' stop bit. All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11 bit. These bit are:

- 8 data bit, least significant bit first.
- 1 stop bit (always 1).
- 1 start bit (always 0).
- 1 parity bit (odd parity).

In this game mouse left and right buttons are used to move the spaceship left and right respectively while the middle button is used to shoot the aliens.

Detailed description of each pin is given in Table 1.

**Table 1**
**Pin Description of Ps/2 Control**

| *PS/2 DIN pin* | *Signal* | *FPGA Pin* |
|---|---|---|
| 1 | DATA (PS2D) | G13 |
| 2 | Reserved | – |
| 3 | GND | GND |
| 4 | Voltage Supply | – |
| 5 | CLK (PS2C) | G14 |
| 6 | Reserved | – |

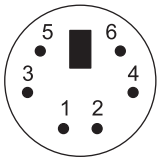


**Figure 2: Male connector**

## B. Game Control

The game control mainly comprises of movement of the spaceship and shooting of the aliens. This requires that when pixels of the missile and the pixels of the aliens are equal, aliens are shot and they are dead. The spaceship only moves left or right and so the aliens. For this we have four states: left, right, up and down.

1. *Method of movement of aliens:* The following code explains the control of aliens:

```
case state_v is
when up =>
 state_v_next <= down;
 master_coord_y_next <= master_coord_y - 4;
 when down =>
 state_v_next <= up;
 master_coord_y_next <= master_coord_y + 4;
```

```
end case;
case state is
when right =>
if master_coord_x + A_WIDTH = 640 then
state_next <= left;
else

when left =>
if master_coord_x = 0 then
state_next <= right;
else
master_coord_x_next <= master_coord_x - 16;
master_coord_x_next <= master_coord_x + 16;
end if;

end if;
end case;
end if;
end process;
```

Here A_width denotes the width of the screen size where aliens should move horizontally. As soon as the pixels reach 640, they restart from the left side.

2. *Method of generating aliens:* The shape of the alien for this game is shown in Figure 3. Aliens have been created by using an array of size 32 × 32. Pixels have been given the value at the place where we want the colour which in turn forms a complete image. The shape above is an image of an alien in this game. '1' indicates that pixel has some RGB value while no value indicates no RGB value. This array is then stored in a ROM memory of FPGA.

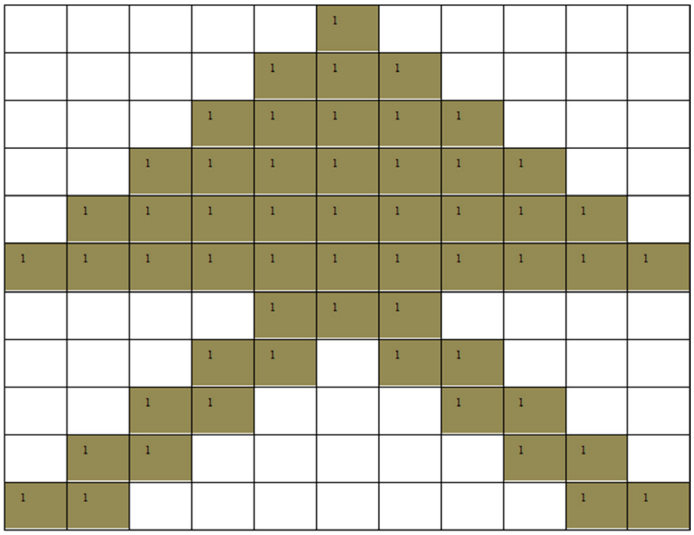

**Figure 3: Graphics of alien**

3. *Method of generating scores:* The following command is used for generation of scores.

```
restart_next <= '1' when defeated1 = '1' and
                         defeated2 = '1' and
                         defeated3 = '1' else
                '0'
level_next <= level + 1 when counter(0) = '0' and
                             defeated1 = '1' and
                             defeated2 = '1' and
                             defeated3 = '1' else
              level;
score_next <= score + 2 when destruction = '1'
                        else score;
```

## C. VGA Module

The function of VGA module in the game is to translate coordinate and pixel settings information received from the graphic display module and the text display module into VGA control signal, and then correctly display the image information by a monitor. A VGA monitor has three colour signals (red, green and blue) that set one of these colours on or off on the screen. The intensity of each of those colours sets the final colour seen on the display. For example, if VGA architecture the red is fully on, but the blue and green off, then the colour would be seen as a strong red. Each analog intensity is defined by a two bit digital word for each colour (e.g. red(0) and red(l) that are connected to a simple digital to analog converter to obtain the correct output signal. The VGA image is controlled by two signals – horizontal sync and vertical sync. The horizontal sync marks the start and finish of a line of pixels with a negative pulse in each case. The vertical sync is similar to the horizontal sync except that, in this case, the negative pulse marks the start and finish of each frame as a whole. The actual image data is sent in a 25.17 µs window in a 31.77 µs space between the sync pulses. The architecture of VGA is shown in Figure 4.

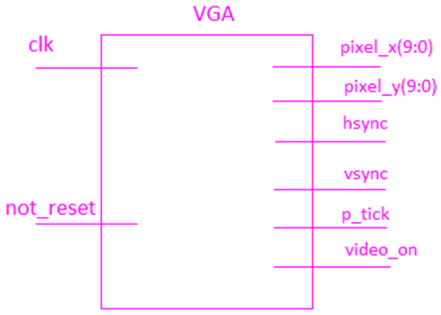


**Figure 4: VGA**

## D. Memory Unit

When we talk about graphics the question that arises is how these graphics are processed and where they are stored. They are stored in the Block Random Access memories (BRAM) and are processed by the GPU. A Graphics Processing Unit (GPU) is a dedicated circuit that operates on a storage area (frame buffer) for the purpose of providing display output. With the increasing graphic requirements in personal, business and embedded

applications, GPUs have become an integral part of most computer architectures. BRAMs are basically fast static RAM bit and each port operates in synchronous fashion. The frame buffer is designed out of BRAM memory and contains 3 ports – two read and one write port. Frame buffer is a memory location which stores the video content that needs to be output for display. One very important design decision is to choose the storage location for frame buffer. Support of 8 colours (3 bit) at a resolution of 320 × 240 require (3/8 bytes × 320 pixels × 240 pixels) = 28800 bytes for image memory.

## 4. DATA PATH ARCHITECTURE

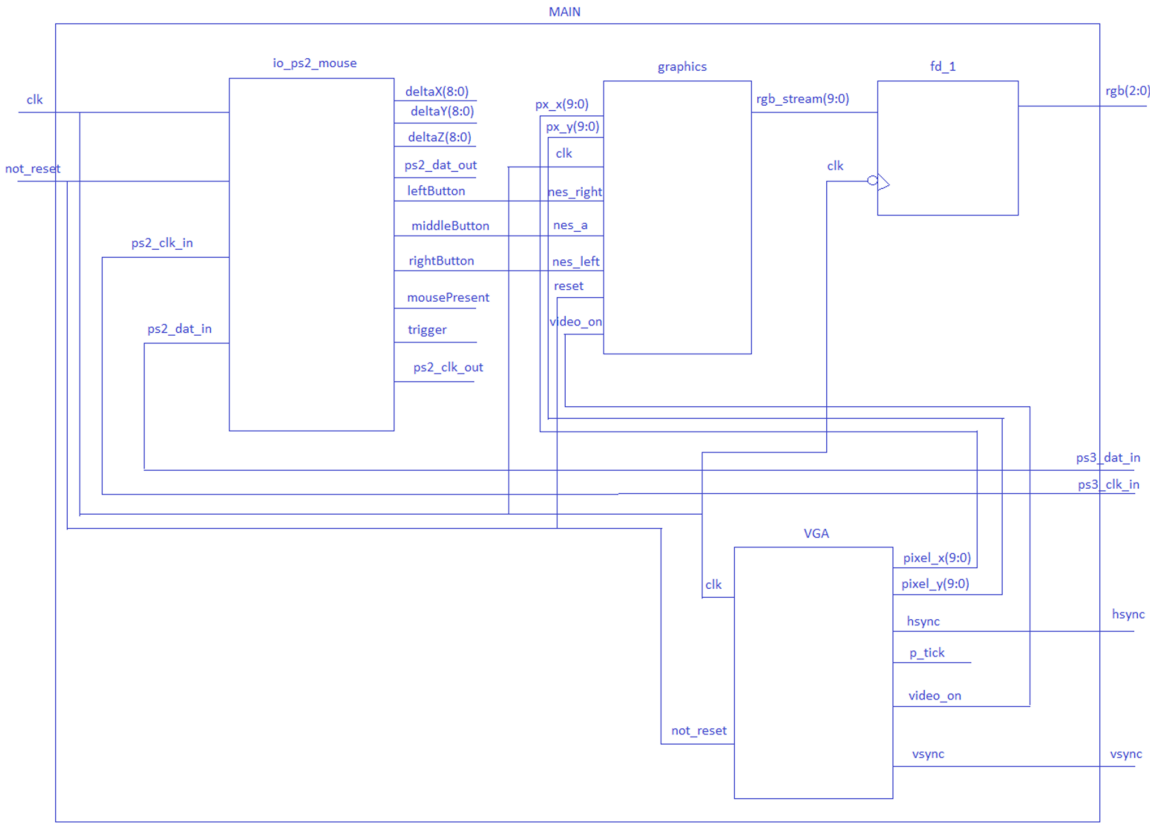


**Figure 5: Data path architecture**

Figure 5 shows the data path architecture of the game space shoot. It shows mainly three modules which are graphics, VGA and io_ps2_mouse. Ps2 has 2 input signals from the board ps3_dat_in and ps3_clk_in which are bidirectional. ps3_dat_in pin is to send the data from the mouse and ps3_clk_in is for the synchronization of the data. Clk is given from the board through C9 oscillator which is providing 50 Mhz and is universal to all the modules and for the synchronization. Ps2 module receives the three bytes of data in deltaX, deltaY and deltaZ where left, right and middle button takes the data from the first 3 bit of the first byte received. This data is actually the status of the three buttons signifying whether the buttons have been pressed or not. These buttons are then connected to the three main input of graphics module nes_a, nes_left and nes_right. The module have already defined the functionality of these signals in the programming that they have to move the spaceship left or right and to shoot the targets. Signal video_on comes from the vga to check if the signal is high or not. Graphics will display only when this signal is one. The output of graphics rgb_stream(2:0) goes to the D flip flop because the the output of this flip flop varies directly according to the input and hence this signal will come out as it is on the output. Whenever the values is changed it is reflected at the output.rgb(2:0) decides the colour of the screen.

The output of the vga are the five pins connected to the board through vga cable i.e hsync, vsync, rgb(0), rgb(1) and rgb(2). The pin configurations of ucf file are shown in the results.

## 5. RESULTS

The result of the developed design for game space shoot is explained with the help of an RTL diagram shown in Figure 6 and corresponding simulated waveform in Figure 7. Detail of the pin is described in Table 2.

**Table 2**
**Pin detail of developed RTL**

| *Pins* | *Description* |
|---|---|
| Clk | This pin provides a synchronous clock to the deisgn |
| Reset | Reset is also synchronized with clock |
| Rgb(2:0) | It provides the colour of the screen |
| Ps3_dat_in | It takes the data from the ps2 and vice versa |
| Ps3_clk_in | This clk is used to synchronise mouse with fpga |
| Hsync | Controls the start and finish of the pixels. |
| Vsync | Controls the start and finish of the whole frame |

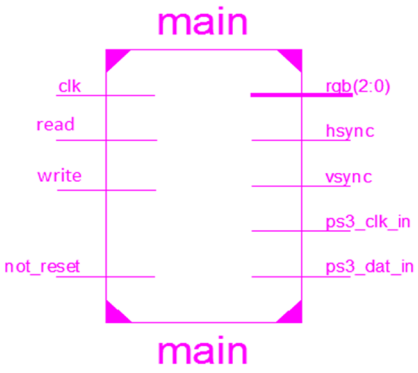


**Figure 6: RTL of Main**

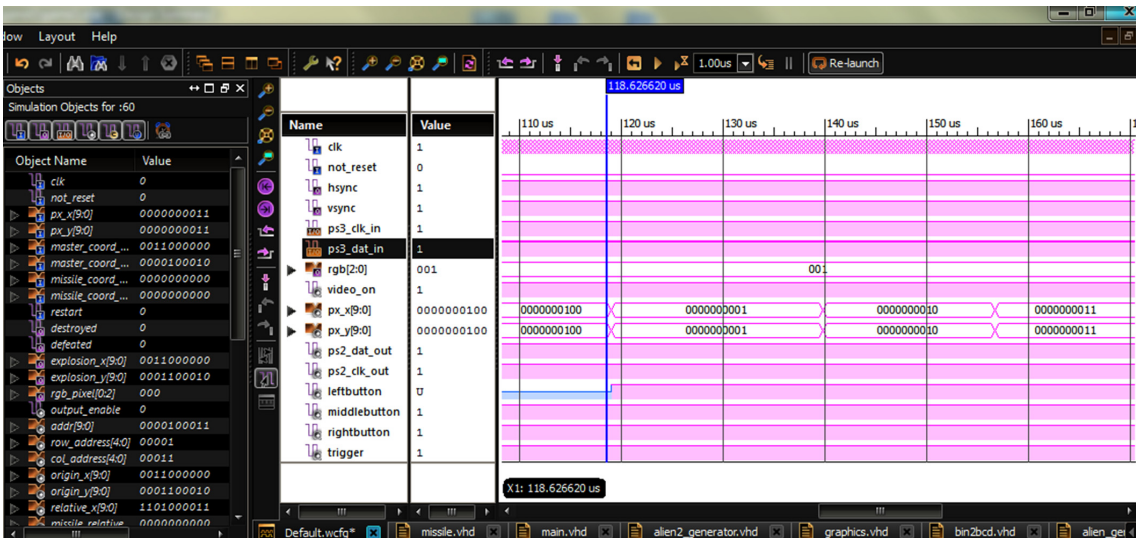


**Figure 7: Simulator waveform**

The above simulation shows the resulting values of the signals. Clock needs to be always high for other operations to take place. video_on is always high as it is set in the program. For the display to occur this signal needs to be set. Graphics always check that if video_on is set only then the further operations in graphics will take place. Clock provided is 50 Mhz with an on time period of 20 ns.

**Test Case:** When clock is set high, video_on becomes high, not_reset = 0. px_x and px_y gets a value when left or right button of the ps/2 is pressed to move the ship. So, px_x and px_y records the next pixel value. RGB can be set to any value from 000 to 111. In this case it is set as 001. The above Figure 6 shows the resulted waveform. px_x(9:0) and px_y(9:0) is 1 when the rightButton and leftButton is pressed. Then both the signal becomes 2, and in the next case 3. This shows that the pixel value gets incremented when left and right button are pressed which means that the movement is done with the increasing pixel value of $x$ and $y$.

## 6. SYNTHESIS

The synthesis work is done on Spartan 3E FPGA and the supporting block diagram is shown in Figure 8. Synthesis block diagram explains that from where the input to the design is coming and where it goes for the further process and control. The main input of this game is ps2 which gives the status of three buttons i.e. left, right and middle in its first byte. This status is received by the graphics module and it allows these signals to move the spaceship. All this is processed by the Xilinx Spartan 3E board. When the data is processed then the output is displayed to the main output which is the monitor screen through the VGA controller. Design summary report is given in Table 3.

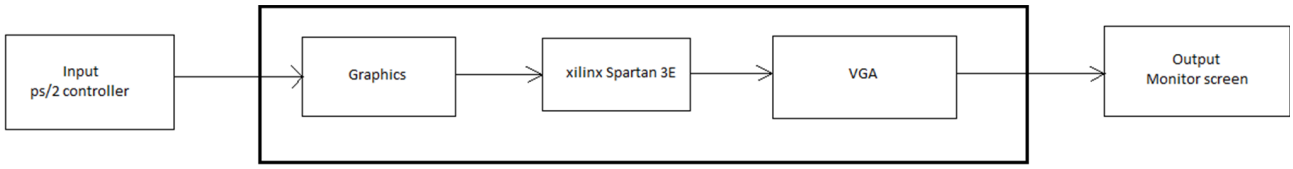


**Figure 8: FPGA Synthesis block diagram**

**Table 3**
**Design Summary and timing parameters**

| *Target device* | *Xc3s500e-4fg320* | *Errors* | *No errors* |
|---|---|---|---|
| Product version | ISE 14.7 | Warnings | 56 warnings |
| Design goal | Balanced | Routing results | All signals completely routed |
| Design strategy | Xilinx Default (unlocked) | Timing constraints | All constraints met |
| Environment | System settings | Final timing score | 0 |
| | *Device Utilization Summary* | | |
| *Logic utilization* | *Used* | *Available* | *Utilization* |
| Number of slice flip flops | 466 | 9312 | 5% |
| Number of 4 input LUTs | 2261 | 9312 | 24% |
| Number of occupied slices | 1368 | 4656 | 29% |
| Number of slices containing only related logics | 1368 | 1368 | 100% |
| Number of slices containing unrelated logics | 0 | 1368 | 0% |
| Total number of 4 input LUTs | 2435 | 9312 | 26% |
| Number used as logic | 2260 | | |
| Number used as a route-thru | 174 | | |
| Number used as shift registers | 1 | | |
| Number of bonded IOBs | 9 | 232 | 3% |
| Number of BUFGMUXs | 1 | 24 | 4% |
| Average fan out of non-clocked nets | 3.83 | | |

The ucf file configurations for vga are as follows:

NET “clk” LOC = “C9”;
NET “hsync” LOC = “F15”;
NET “vsync” LOC = “F14”;
NET “rgb<0>” LOC = “G15”;
NET “rgb<1>” LOC = “H15”;
NET “rgb<2>” LOC = “H14”;
NET “ps3_dat_out” LOC = “G13”;
NET “ps3_clk_out” LOC = “G14”;

## A. Experimental Setup

The experimental setup is shown in Figure 9. It shows two main desktop screens first which has an IDE for programming the game which shows the graphics output and the second shows the graphics output. First desktop is connected to the board with the help of programming cable that helps to burn the bit file in to the FPGA and the second desktop is connected to the board through VGA cable. The clear graphics for alien and spaceship can be seen in Figure 10.

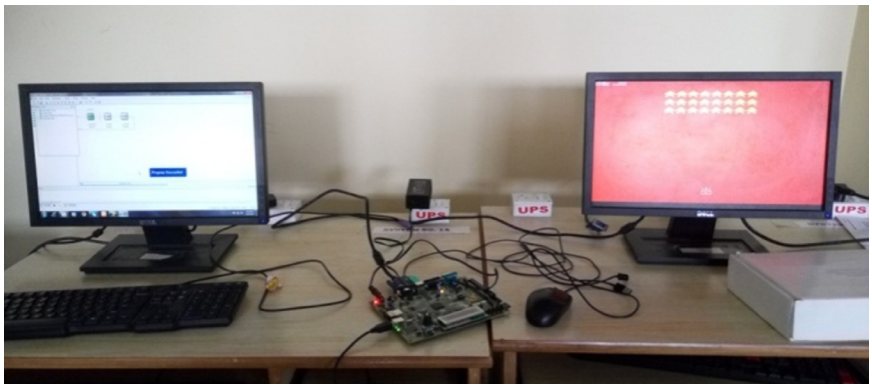


**Figure 9: Experimental set up**

**Figure 10: Graphics of alien**

## 7. CONCLUSION

The game is programmed in ISE Xilinx 14.7 and is simulated in ISim. Problems did occur in interfacing the PS/2 controller with FPGA since the controller used a different clock than what is given to the board. Also PS/2 receives and sends data in an 8 bit frame with a start bit and a stop bit, so this requires a bit heavy time for the interfacing of PS/2 with FPGA. It is a good choice to create it on FPGA, because we had a suitable development board which has VGA port for interfacing a computer monitor and ps/2 for controls. Making it on FPGA is a good idea also, because this way we had an embedded system, we don't have other dependencies not even operating system dependency. We also chose this platform, because this way we could create an ASIC, a standalone chip, by converting the VHDL code to Verilog with Mentor Graphic tools and we could create the layout of the chip for the final product. The layout will be made on very big number layers and as in integrated circuits is done, will be done with auto route. This way we had a game implemented on a chip, this means that we have a standalone device with no speed issues, which would have been there if it would be made on a microcontroller because microcontroller processes small amount of data. And video processing type of processes cannot be handled easily in controllers because of less memory space and timing cycle constraints.

## REFERENCES

[1] B. Bowman, N. Elmqvist, T.J. Jankun-Kelly, "Toward Visualization for Games: Theory, Design Space, and Patterns" IEEE transactions on visualization and computer graphics, Vol. xx, 2012.

[2] CHEN Xi, "Analysis and Application of PS/2 Device Interface Protocol", CNKI Journal, Electronic Design Engineering 2004 -04, retrieved from http://en.cnki.com.cn/Journal_en/I-I135-GWDZ-2004- 04.htm

[3] G. Wang, T. N. Tran, H. A. Andrade, "A Graphical Programming and Design Environment for FPGA-based Hardware" IEEE, pp (337-340), 2010.

[4] H. F. Jimenez, R. F. M. Gonzalez, P. V. G. Hernandez, A. D. Cedillo, "Catching LED" Game: An FPGA Developing Board Implementation" International Journal of Computer and Information Technology, Volume 04, pp (258-262), 2015.

[5] J. Qu, Y. Wei, Y. Song, "Design Patterns Applied for Networked First Person Shooting Game Programming" IEEE, 2014.

[6] K. Liu, Y. Yang, Y. Zhu, "Tetris game design based on the FPGA" IEEE, pp (2925-2927), 2012.

[7] M. Boule, Z. Zilic, "An FPGA Based Move Generator for the Game of Chess" IEEE custom integrated circuits conference, pp (71-74), 2002.

[8] M. M. Zavlanos, G. J. Pappas, "Distributed Hybrid Control for Multiple-Pursuer Multiple-Evader Games" University of Pennsylvania.

[9] N. Zhar, M. A. Ali, M. Eleuldj, A. Raji, "A Specific-domain Design Tool for FPGA-based Image and Video Processing System" *International Journal of Computer Applications, Volume 56, pp(16-21), 2012.*

[10] P. K. Gaikwad, "Development of FPGA based PS/2 Mouse and VGA Monitor Interface Technique" International Journal of Research in Engineering & Advanced Technology, Volume 1, 2013.

[11] P. Lankoski, S. Bjork, "Gameplay Design Patterns for Believable Non-Player Characters" Proceedings of DiGRA 2007 Conference, pp (416-423), 2007.

[12] P. P. Chu, "*FPGA Prototyping by VHDL Examples" Ch-9,* John Wiley & Sons, Inc, pp (199-214), 2008.

[13] R. Drechsler, N. Drechsler, "GAME-HDL: Implementation of Evolutionary Algorithms Using Hardware Description Languages" Springer-Verlag Berlin Heidelberg, pp (378–387), 2003.

[14] R. J. Teather, J. Carette, M. Thevathasan, "Uniform vs. Non-Uniform Scaling of Shooter Games on Large Displays" McMaster University Hamilton, ON, Canada.

[15] R. J. Teather, M. Thevathasan, J. Carette, "Scale Effects in "Bullet Hell" Games" McMaster University Hamilton, ON, Canada.

[16] R. P. McMahan, E. D. Ragan, A. Leal, R. J. Beaton, D. A. Bowman, "Considerations for the use of commercial video games in controlled experiments" International Federation for Information Processing Published by Elsevier B.V, pp (3-9), 2011.

[17] R. Szabó, A. Gontean, "Pong game on an fpga development board using a computer screen as display" International Journal of Computer Science and Applications, Vol. 12, pp. 70 – 80, 2015.

[18] S. Gao, S. N. Givigi, A. JG Beaulieu, "Fpga implementation of multiple pursuit-evasion games with decentralized learning automata" IEEE, 2014.

[19] S. N. Givigi Jr, H. M. Schwartz, "Decentralized Strategy Selection with Learning Automata for Multiple Pursuer-Evader Games" Carleton University.

[20] S. Zafar, S. Kataria, A. Sharma, "Digital design of a dedicated Graphics Processing Unit (GPU) architecture for microcontrollers" International Conference on Electronics and Communication System (lCECS), 2014.

[21] V. Alves, I. Cardim, H. Vital, P. Sampaio, A. Damasceno, P. Borba, "Comparative Analysis of Porting Strategies in J2ME Games" Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), 2005.