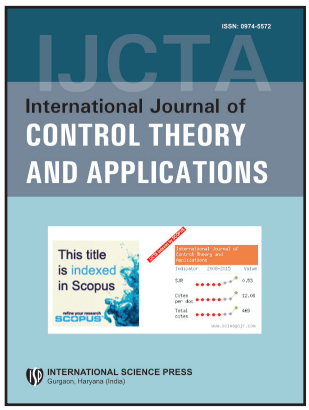

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 9 • Number 44 • 2016

Design and Implementation of a FPGA based Game "Space Invaders"

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

[a,c]M.tech scholar, Embedded System with specialization in wearable technology, University of Petroleum & Enegy Studies, Dehradune, India. Email: [a]rsparihar10@gmail.com; [c]akarsharr1@gmail.com
[b]Department of Electronics, Instrumentation and Control, University of Petroleum and Energy Studies, Dehradun, India. Email: adeshkumar@ddn.upes.ac.in

Abstract: The paper present the design and implementation of the game "Space Invader" taking its inspiration from game "Star Fox", a space shooter game for the Super Nintendo Entertainment System.. The game is all about shooting stars in the sky by moving the space ship in the left and right direction with the help of push buttons of the boards. The score of the game keep increasing as we shoot the stars and once all the stars are shot, the level of the game changes. The approach of designing a project was modular in nature, attempting to create modules starting from display through a VGA controller then designing graphics for aliens and spaceships and creating logics for lives, scores, shooting and shifting of spaceship. The game is made using the software Xilinx ISE 14.7 and is synthesized on the board Xilinx Spartan 6.

Keywords: FPGA, VHDL, Spartan 6, VGA.

1. INTRODUCTION

When the world faced a high economic crisis [7], the gaming market was the one with 30 to 40 percent of growth rate. The biggest advantage of this sector is that it works irrespective of age and taste factor of the user. In today's era, gaming market is expanding with the same ratio as the number of android user.

The main objective of the paper is to present the FPGA methodology used to fulfil technical and motivational requirements[12]. The implementation of a video-game such as "Space Invaders" in hardware not only meets technical design goals, but also highly motivates in comparison to other type of projects, which is a key aspect in the learning process. Visualisation of graphics on the screen is the prime aim of the project [2], which can be implemented using an FPGA board Xilinx Spartan 6.The goal was to re-create the game entirely on an FPGA.

The creation of the video-game is a challenge related to a reality that is well known compared to other possible hardware projects. The approach of the game design is modular in nature[8]. The game starts with the visualisation of the graphics such as stars and spaceship on the screen for the interactive player [1,11]. In order to achieve the goal a VGA [17] interface is used as this is one of the most popular displaying interface. The VGA

controller uses five signals i.e. red, green, blue, horizontal synchronization (hsync) and vertical synchronization (vsync). The first three signals are for colour generation of each pixel at a given location and the later two are there to control the timing of the scan rates. The RGB signals are analog in nature whereas hsync and vsync are digital in nature. The graphics are stored in the RAM of the device and the graphics are processed by 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. The next task of the game is to control the game using the push buttons of the board Spartan 6. In order to do so, the counter clock division technique is used. The clock is divided so that the ship could move bit fast as compared with the provided clock and a counter is set. The rules of the game are simple. The "Space Invader" game is all about shooting stars in the sky and gaining points. We can move the ship in the left and the right direction by using the P4 and F6 push buttons of the board and N4 for the shooting purpose. Every time user shoot the star, the score of the game gets updated and when all the star shot the level of the game changes. There is no death condition for the spaceship. The game continues as the star keeps appearing on the screen. The reason of designing such a simple game was to get a learning experience on FPGA with some real working product.

2. RELATED WORK

The work relating to designing the game on FPGA has been done earlier by many researchers. A. Drachen et. at [1] in their journal have explained the case studies which showcase new ways to analyze gameplay metrics and take advantage of the spatial dimension of certain metrics in order to target gameplay analysis. B. Bowman, et. at [2] have proceed to use the framework to identify common design patterns for how to employ visualization in computer and video games. C. Harrison, et. at[3] in their journal teaches low-level C programming and VHDL coding to design and implement a project of their own choosing. D. Teger, et. at [4] in their thesis work have developed a game "Duck Feed" is implemented entirely on an Altera FPGA utilizing the Nios II microprocessor and custom hardware peripherals. Dr. S.J. Hsieh, et. at [5] in his paper has described the development of a game to help students to learn about industrial wiring of an automated system..

E. Aarseth, et. at [6] the paper seeks to outline and promote a methodology for the aesthetic study of games, which, given the current nascent state of the field, will doubtless give way to more sophisticated approaches in the years to come. H. Ezgi, et. at [7] in their master thesis developed a game "Humankillers" which is a 3D single player first-person shooter game creating using C# in XNA game development studio. H. F. Jimenez, et. at [8], the work is about designing and implementation of a game kind "Simon says". Game system was divided, division provides advantages. J. Qu, et. at[9] discussed the applying of design patterns in First Person Shooting game development, provide solutions to general object-oriented game development in software engineering development process with increased maintainability, reusability and lower risk to adapt the changing requirements and core game mechanism in game development. J. Ha Lee, et. at [10] present a faceted classification scheme for video game genre based on the analysis of hundreds of pre-existing genre labels collected from existing video game organization systems.

L. Fan, Y, et. at [11] in their project report has design a game which will be implemented with System Verilog and C language and shown on the screen through VGA. M. Sanchez-Elez1, et. at [12] in their paper presented the project based learning (PBL) methodology used to fulfil both technical and motivational objective. O. Dahlberg, H. Ericsson, J. Hasselqvist, A. Josefsson, H. Ottervad, et. at [13] the main focus for the thesis is to present the graphical effects used to create a graphically intensive game within a limited time frame, but it also presents the different solutions chosen to create the game engine used by the game. R. J. Teather, et. at [14] presented a study comparing player performance in a shooter game using two different types of scaling across four display sizes. The first scaling type used uniform scaling where increasing the display size also increased

the size of all in-game elements by the same factor. The second employed non-uniform scaling where all in-game elements remained fixed in size, but the game environment increased (or decreased) in size. R. Szabo, et. at [15] in his paper had the idea to create the chip placement game on a chess table. The chips placement is done on a chess table, where you can place a number of four to eight chips on a 4×4 to 8×8 chess table.

R. Szabo et. at [16] the paper presents the creation of the Pong game running on an FPGA with a computer display connected to it. The game is optimized to work on both LCD and CRT displays too. R. Wasu1, et. at [17] we take the programming methodology and adopt the integration tools (Quartus version 13.0 of Altera). S. Xiao, et. at [18] the purpose of the project is to recreate, on a Xilinx Field Programmable Gate Array (FPGA), the classic version of 'Asteroids' by programming hardware functionality using the verilog description language. U. Reinsalu, et. at [19] the paper gives an overview of the courses taught for IT students with rather different background. Z.Nizam, et. at [20] the aim of this project is to formulate this concept (B3/S23) in to a visualization can be implemented using an FPGA board.

3. SYSTEM BLOCK DIAGRAM

The block diagram of the game Space Invader is shown in Figure 1. The entire system is divided into two main blocks named Game logic and Game control developed on the platform Spartan 6.The Game logic is basically responsible for designing and positioning of the graphics and displaying it on the screen. The Game control logic is responsible how the game should be played. In game logic, under graphics module the graphics of the stars and spaceship is created and their positioning is decided. For the game control, the push buttons (P4, F6, N4) of the Spartan 6 board are used.

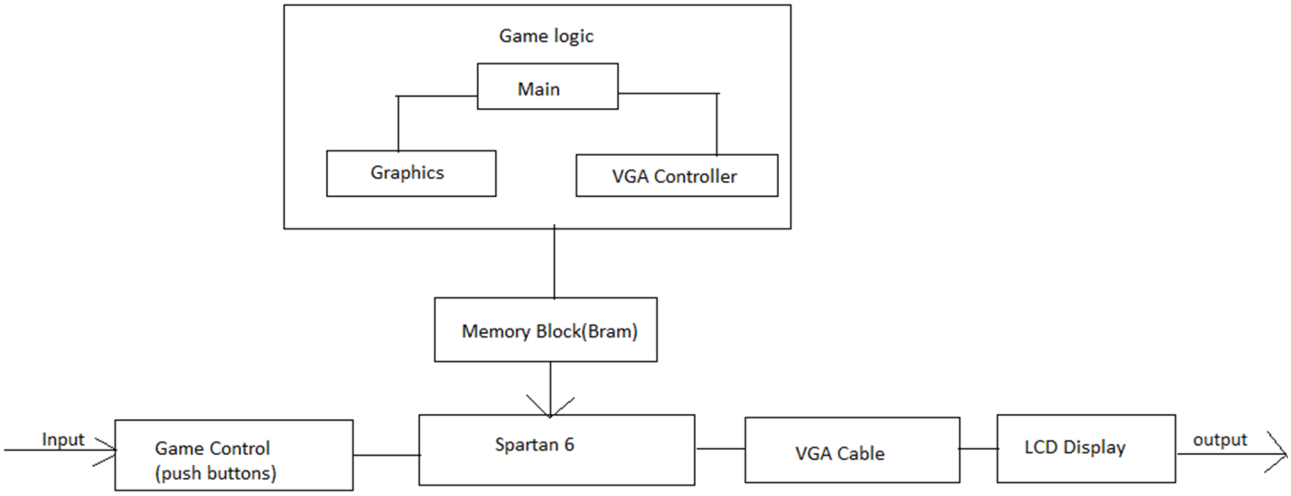

Figure 1: System block diagram of the game

A. Game Logic

1. *VGA controller* – The main objective of the VGA controller is to generate display by visualizing the cellular patterns created in each steps. The screen used is a normal 640×480 desktop (or laptop) display unit (14 inch) to show the output at about 50 MHz.

   We need the pixel clock of VGA controller to be at 25.175 MHz (which we can generate from 100MHz system clock). An important fact to realize is that the VGA monitor does not have memory and thus will not store the pixel information being written to it. Instead, the pixels must be continuously sent to the display to achieve a stable image.

The VGA controller needs to be receiving from the memory pixels and will be responsible for constantly sending out the pixel information. VGA is an analogue video standard that is mostly used in personal computers. VGA can also refer to a piece of display hardware developed by IBM (Video Graphics Array) or a display mode, that uses 640 × 480 pixels resolution.

VGA connector uses a total of 15 pins, but only 5 signals are needed for operation:

- HSYNC – horizontal synchronization signal for the horizontal position of the active pixel.
- VSYNC – vertical synchronization signal for the vertical position of the active pixel.
- RED – red colour channel
- GREEN – green colour channel
- BLUE – blue colour channel

RGB signals are analog in nature whereas hsync and vsync signals are digital in nature.

2. *Graphics* – For designing of graphics a matrix of 32 × 32 is created where a 3-bit data is given defining the RGB values of the pixel. The position where the values are given defines the shape of the graphics.

The Table 1 shows the 3 bit colour coding ranging from 000 to 111.The Figure 2 shows a matrix for forming the graphics. The placement of the colour code is deciding the shape and code decide the colour of the graphics. The Figure 3 shows the actual graphics of the stars used in the game.

Table 1
3-bit colour coding

000	Black
001	Blue
010	Green
011	Magenta
100	Red
101	Purple
110	Yellow
111	White

							100							
						100	100	100						
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	100	100	100	100	100	100	100	100	100	100	100	100	100	
		100	100	100	100	100	100	100	100	100	100	100		
		100	100	100	100	100	100	100	100	100	100	100		
	100	100	100	100	100	100	100	100	100	100	100	100	100	
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
						100	100	100						
							100							

Figure 2: Pixel matrix for graphics creation

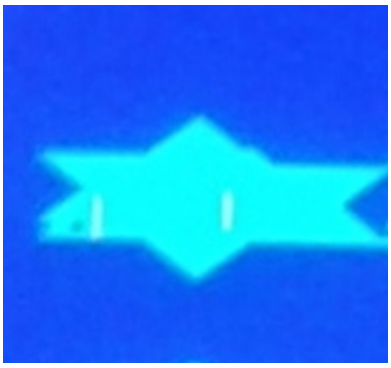

Figure 3: Star graphics of the game

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;
```

B. Game Control

The game is being controlled by the push buttons of the board. The ship can move in the left and right direction by using the P4 and F6 buttons of the board respectively. The main logic used to move a ship is the counter clock division method. A counter is set to count the width of the screen both the sides and do not allow the ship to move further out of the screen. The clock has been divided to control the movement of the ship and make it move fast.

Counter and Clock Division

```
divide <= (others => '0') when not_reset = '0' else sig2 when rising_edge(clk);
sig1 <= std_logic_vector((counter)+2) when compare = '1' else counter;
sig2 <= std_logic_vector((divide)+2) when compare = '0' else (others => '0');
compare<='0'whendivide<
"10111110101111000010000000"      else '1';
counter <= sig1 when rising_edge(clk);
```

Buttons

sigXValue <= std_logic_vector((position_next)+5) when nes_right = '1' and compare = '1' else

std_logic_vector((position_next)-5) when nes_left = '1' and compare = '1' else position_next;

position_next <= sigXValue when rising_edge(clk)

C. Memory Unit

The graphics 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

The inside structure of the "Space Invader" can be seen in the Figure 4 which shows the data path architecture. As we can see that the inside structure is quite simple for our circuit. We have only 3 chips inside, one for the VGA sync and one for the graph part for drawing the stars and spaceship. The last chip is a D-latch for creating a delay for timing.

The entire logic of the game lies in the prime frame called "main". The main has two sub-parts called graphics and VGA from where it takes all logic from. The entire logic is port mapped into main from where the game actually take charge of.

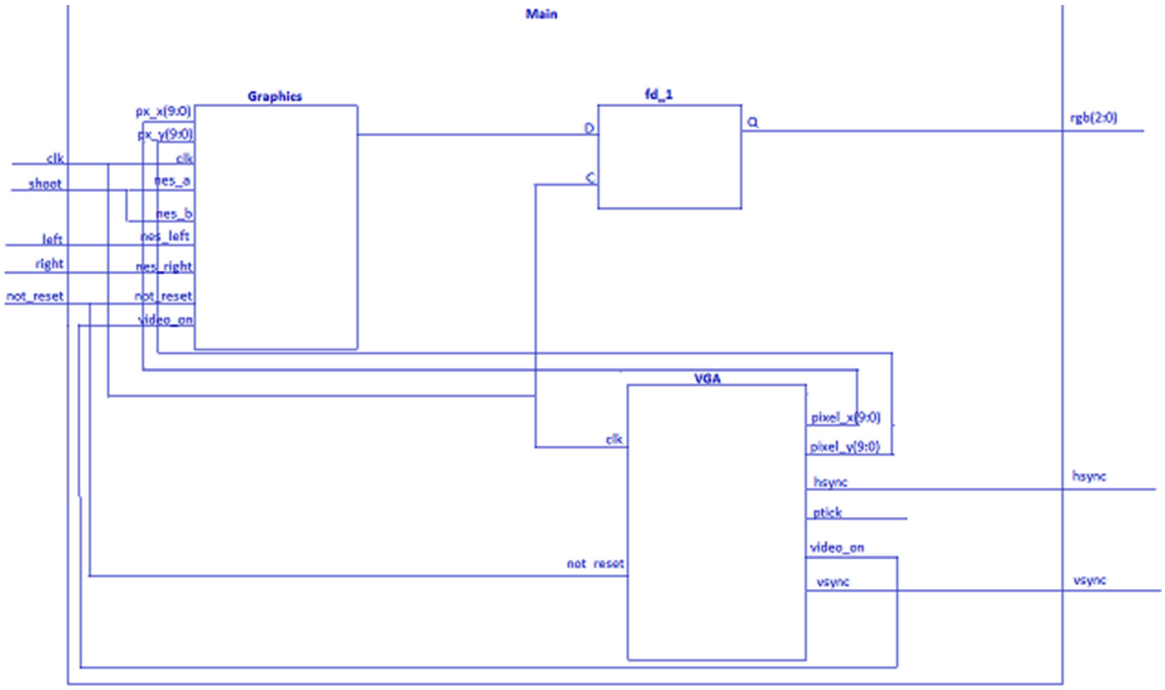

Figure 4: Data path architecture

The clock signal is used to synchronize graphics and the VGA module. The not_reset signal is used to reset the entire circuitry with the reverse logic. The video_on signal of the graphics should be synchronized with the video_on signal of VGA to produce the graphics. The graphics has px_x and px_y for the placement of the ship and the same signal goes to VGA module in order to visualize the ship position. The nes_a and nes_b signals are used to produce the shooting signal. The nes_left and nes_right signal are responsible for the movement of ship in the left and right direction. The graphics produce a 3 bit rgb signal to get the display and VGA produce Hsync and Vsync signal to create a horizontal and the vertical porch of the screen.

5. RESULT AND DISCUSSION

The result of the developed design for game space invader *t* is explained with the help of an RTL diagram shown in Figure 5 and corresponding simulated waveform in Figure 6. Detail of the pin is described in Table 2.

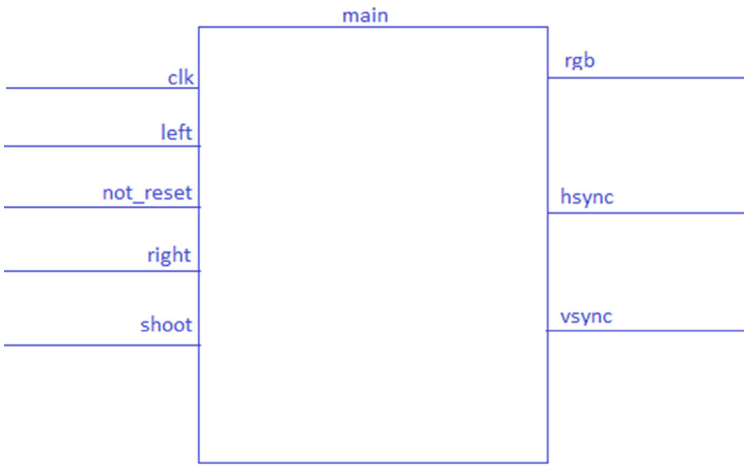

Figure 5: RTL view of the game

Table 2
Pin description of the RTL

clk	A clock of 50 Mhz is used to synchronize the VGA and graphics with the main
Left	An input pin defined to move the ship to the left direction with the help of push buttons of the board
Right	An input pin defined to move the ship to the right direction with the help of push buttons of the board
Shoot	An input pin defined to shoot the stars with the help of push buttons of the board
Not_reset	Is the reverse logic defined to reset the entire game.
RGB	A 3 bit data pin used to produce graphics on the pin
Hsync	Horizontal synchronization for VGA controller
Vsync	Vertical synchronization for VGA controller

The given Table 3 shows the design summary of the entire game implemented on Xilinx ISE software. Here is more information about the VHDL code, errors, warnings, used logic structures and the FPGA usage percentage.

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*.

The 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 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 1: 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 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.

Test case 2: When nes_a and nes_b signal gets high the shoot signal becomes 1.

Test case 3: When shoot = 1 then signal destruction checks its value and goes high if occurred.

Test case 4: If destruction = 1, score signal gets updated by increasing the value by 2.

Test case 5: If defeated 1,defeated 2 and defeated 3 signal are all 1 and counter = 0, then level= level +1.

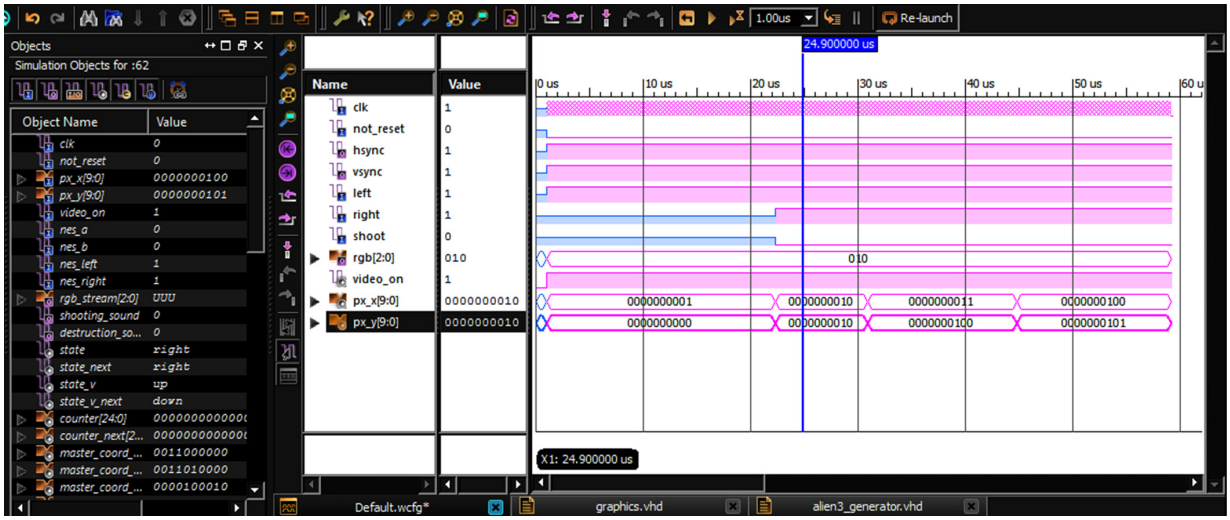

Figure 6: Timing signal diagram

Table 3
Device utilization summary

Logic utilization	*Used*	*Available*	*Utilization*
Number of slice flip flops	371	9312	3%
Number of 4 input LUTs	1902	9312	20%
Number of occupied slices	1133	4656	24%
Number of slices containing only related logics	1133	1133	100%
Number of slices containing unrelated logics	0	1133	0%
Total number of 4 input LUTs	2060	9312	22%
Number used as logic	1902		
Number used as a route-thru	158		
Number used as shift registers	1		
Number of bonded IOBs	10	232	4%
Number of BUFGMUXs	1	24	4%
Average fan out of non-clocked nets	3.90		

6. SYNTHESIS

The synthesis work is done on Spartan 6 FPGA and the supporting block diagram is shown in Figure 8.The synthesis diagram explains the entire process of occurrence. The game take its input from the push buttons of the board which can move the ship in the left and right direction and can shoot the stars. The signal produce are generally analog in nature which need to be converted to digital form. The entire process takes place in the graphics module. Now these signals are again converted to analog form in order to get the display through a VGA. The entire process is done on the platform Spartan 6.

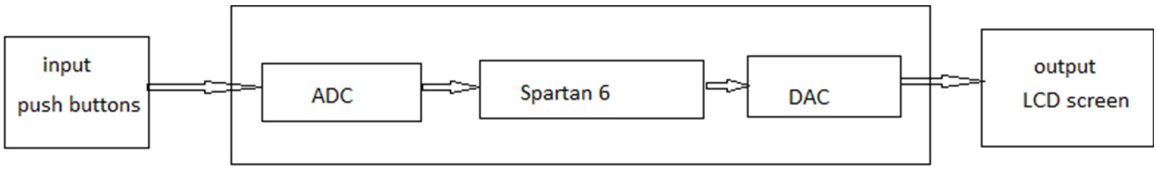

Figure 7: FPGA synthesis process

7. EXPERIMENTAL SET UP

The clear graphics of the stars and spaceship can be seen in Figure 8. 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.

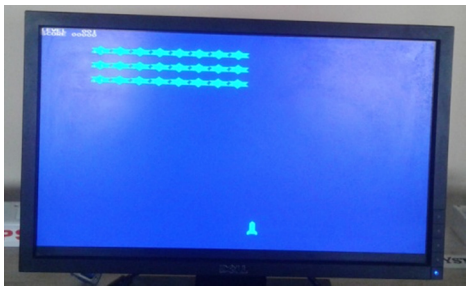

Figure 8: Game view

Figure 9: Experimental set up

8. CONCLUSION

The main goal of the project was to reproduce one of the entertaining games with great learning objective. The entire game is developed using a FPGA platform which seems to be a good choice because we had a suitable development board which has VGA port for interfacing a computer screen with the push buttons for controls. Making it on FPGA was a good idea also, because it had an embedded system, this way we don't have other dependencies not even operating system dependency. The problem of speed and memory are also resolved as FPGA is a system on chip and supports projects with high data processing as compared to microcontroller.

For future plan to extend the controls from push buttons to mouse and keyboard on PS/2 interface. The project could be port to other platform too, like the ATLYS board, and create the video drivers on DVI or HDMI interfaces, for newer monitor types and create the keyboard and mouse drivers on USB interface or even add joystick drivers on USB interface.

Acknowledgment

It is a pleasure to acknowledge " Electronics, Instrumentation and Control Department", for giving the opportunity and allowing to do our project in the labs. We sincerely convey our gratitude to our project guide Dr. Adesh Kumar whose continuous support andencouragement helped to complete the project. We are also thankful to all the faculty and supporting staff who provided us with adequate data information and help in spite of their hectic schedule.

REFERENCES

[1] A. Drachen and A. Canossa, *"Towards Gameplay Analysis via Gameplay Metrics"*, journal from MindTrek, pp(202-209), 2009.

[2] B. Bowman, N. Elmqvist, and T.J. Jankun-Kelly, *"Toward Visualization for Games:Theory, Design Space, and Patterns*", IEEE transaction on visualization and computer, pp(1-14), 2012.

[3] C. Harrison and P. Jones, *"Experiences with FPGA teaching," in The Teaching of Digital Systems,* IEEE Colloquium on, 1998.

[4] D. Teger, S. Rogowski, J.Dinerman, K and Ramkishun, *"DuckFeed: An Embedded Take on Duck Hunt*", a thesis report on Embedded system design from Columbia University, pp(1-51), 2011.

[5] Dr. S.J. Hsieh, Texas A&M University, "*Use of games for learning automated system intergration*", journal presented in American Society for Engineering Education, 2012.

[6] E. Aarseth, *"Playing Research: Methodological approaches to game analysis*", Papers from spilforskning.dk, University of Bergen, pp (1-7), 2014.

[7] H. Ezgi, Tuglu and K. Akyil, "*Design and implementation of a single-player first-person shooter game using XNA game development studio"*, master of science thesis in the department of computer Science and Engineering, Chalmers University of Technology, pp(1-96), 2010.

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

[9] J. Qu, Y. Wei and Y. Song, *"Design Patterns Applied for Networked First Person Shooting Game Programming"* Department of Computer Science & Information Technology, Clayton State University, Morrow, 2014.

[10] J. Ha Lee, N. Karlova, R. Ivy Clarke, K. Thornton and A. Perti, *" Facet Analysis of Video Game Genres"*, iconference, pp(126-139), 2014.

[11] L. Fan, Y. Wang, D. Yang, and Y. Zhu, *"Battle Tank- Inspired Multidirectional Shooter Video Game*", Embedded System design, Columbia University, 2014.

[12] M. Sanchez-Elez1 and S. Roman, "*Learning Hardware Design by implementing student's Video-Game on a FPGA''*, Int'l Conf. Frontiers in Education: CS and CE, pp(25-30), 2015..

[13] O. Dahlberg, H. Ericsson, J. Hasselqvist, A. Josefsson, H. Ottervad, "*A Case Study of a 3D Space-Shooter Game*", Bachelor of Science Thesis in Computer Science and Engineering, CHALMERS UNIVERSITY OF TECHNOLOGY, pp(1-47), 2015.

[14] R. J. Teather, J.Carette, M Thevathasan, "*Uniform vs. Non-Uniform Scaling of Shooter Games on Large Displays*", 2015.

[15] R. Szabo and A. Gontean, "Pong game on an FPGA development board using a computer scree display", International Journal of Computer Science and Applications, pp(70-80), 2015.

[16] R.Szabo, "*Creation of the Chips Placement Game with Backtracking Method in Borland Pascal''*, IEEE journal on Applied Electronics, 2014.

[17] R. Wasu1 and V. R. Wadhankar, "*Review on Design of VGA Controller Using FPGA*", International Journal of Science and Research, pp (1734-1737),2013.

[18] S. Xiao and J. Verrill, "*Creation of Asteroids Game Using Verilog and Xilinx FPGA*", a final project thesis on Embedded system design, pp(2-85), 2015.

[19] U. Reinsalu, A. Arhipov, T. Evartson, and P. Ellervee, "*HDL-s for Students with Different Background*" in Microelectronic Systems Education,2007. IEEE International Conference, 2007.

[20] Z.Nizam, "*FPGA based Game of Life*", project report from Digital System Design,Department of Electronic and Telecommunication, University of Moratuwa, pp(2-21), 2014.