International Journal of Computer Science and Applications,
©Technomathematics Research Foundation
Vol. 12, No. 2, pp. 70 – 80, 2015

PONG GAME ON AN FPGA DEVELOPMENT BOARD USING A COMPUTER SCREEN AS DISPLAY

ROLAND SZABÓ

Applied Electronics Department, Faculty of Electronics and Telecommunications, Politehnica University Timişoara, Vasile Pârvan Av., No. 2
Timişara, 300223, România
roland.szabo@etc.upt.ro

AUREL GONTEAN

Applied Electronics Department, Faculty of Electronics and Telecommunications, Politehnica University Timişoara, Vasile Pârvan Av., No. 2
Timişara, 300223, România
aurel.gontean@upt.ro

This 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. The game is implemented from scratch, it has the two pads (one read and one blue) drawn on the screen, the green ball is also drawn and the background is painted in white. Everything is implemented in hardware on an FPGA, so this way at the end we are able to convert the implementation to create an ASIC with the Pong game. The goal is to create the Pong game on a single chip.

Keywords: ASIC, chip, computer display, CRT, FPGA, LCD, Pong game, push button.

1. Introduction

The creation of games was always in the mind of the engineers. From the start of engineering scientists always thought about how to create something to entertain themselves, not only the required research work.

The beginning of the electronics gave big opportunities to the engineers to create games. The first games were also blinking lights and LEDs, but the real games could be made after the introduction of the displays.

One of the first games created was the predecessor of the Pong game, which original name is Tennis For Two, developed on oscilloscope and created by William Higinbotham in 1958.

On oscilloscope there were created two paddles and a ball, which could be hit by the paddles, if the ball was missed, the other player got a point.

We wanted to reproduce the history, but on FPGA and after on a standalone chip.

We wanted to recreate the Pong game from scratch, by drawing the paddles and the ball, panting the background and displaying it on a computer display.

We chose this game, because it one of the big classic games and it needs not so much drawing, but it’s really much fun to play.

2. Problem Formulation

Our task was clear and simple; we needed to refresh the history by recreating one of the pioneers of the computer gaming history, the Pong game.

To make it a little more difficult and interesting, we set the task to create the game on FPGA, because in the future we plan to create the stand alone ASIC, the chip with the Pong game.

The hardware what we had was the NEXYS 2 development board (Fig. 1) with Spartan-3E FPGA (Fig. 2). With this we had also other hardware that we could use like a CRT or LCD computer display to be able to have the whole computer game system. A block diagram of the experimental setup can be seen on Fig. 3, we can see that all the Pong game is on the FPGA board which is connected via VGA port to a PC monitor. The game could be controlled with mouse, keyboard or as in our implementation with the 4 push buttons from the FPGA board.

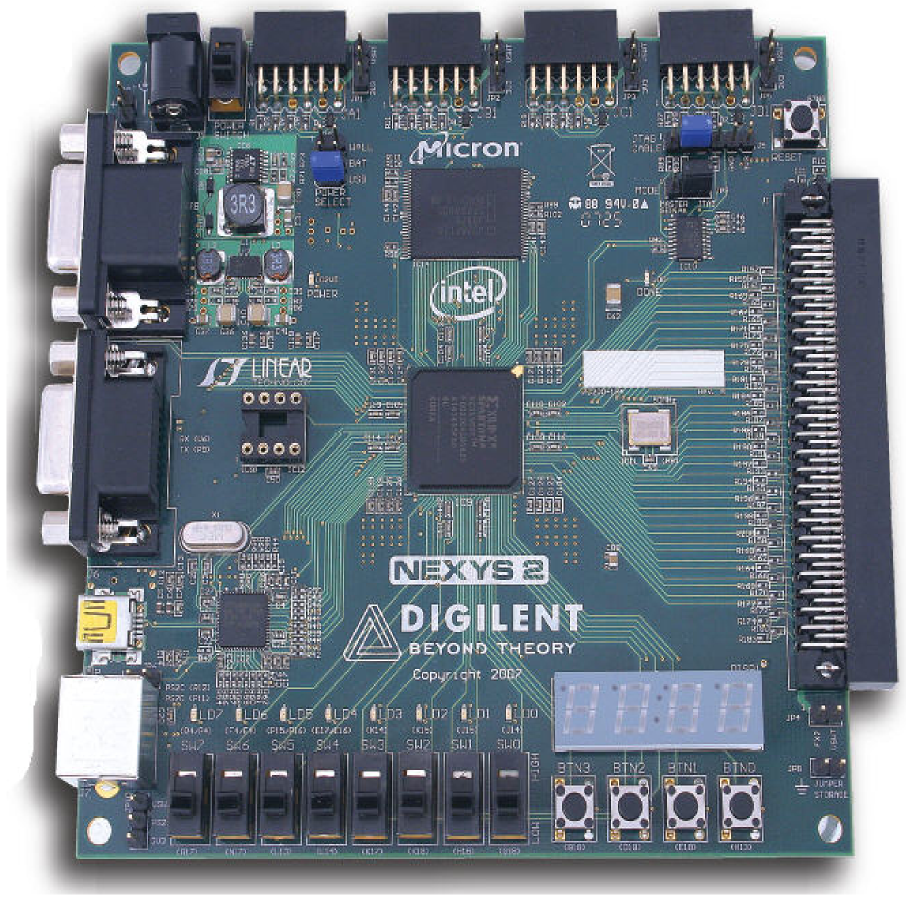


Fig. 1. The NEXYS 2 development board from Digilent with Spartan-3E FPGA.

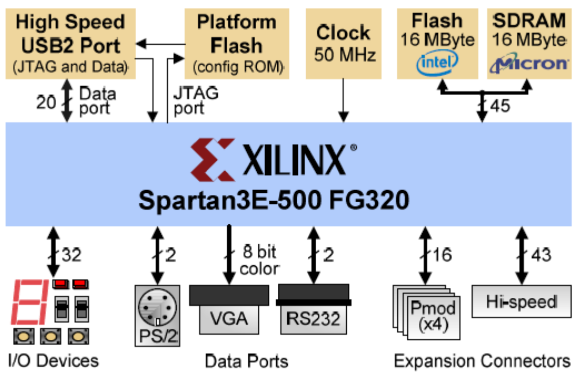


Fig. 2. The Spartan-3E FPGA from the NEXYS 2 development board.

3. Problem Solving

3.1. *Theoretical Background*

We needed to study the whole structure of the development board in order to be able to create the game and to be able to build the correct driver for the VGA monitor, even the Spartan-3E pins.

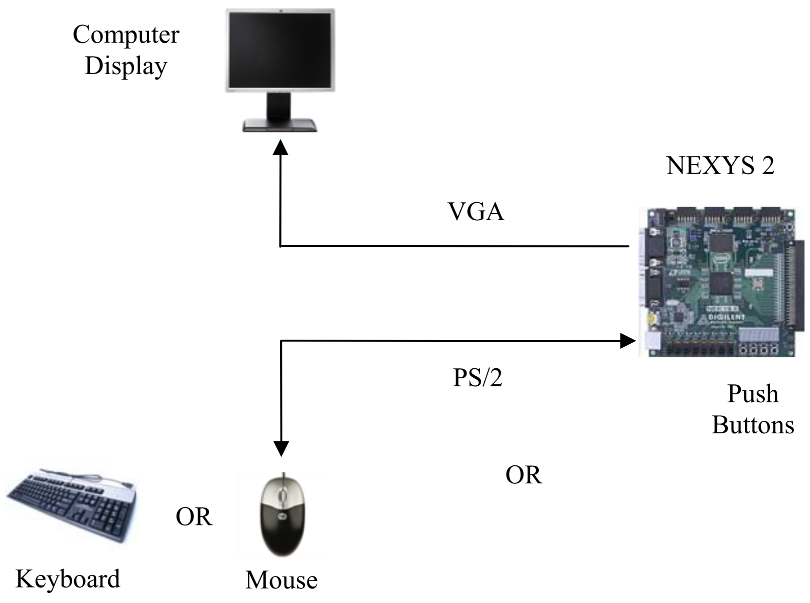


Fig. 3. The block diagram of the experimental setup.

On Fig. 4 we can see the Spartan-3E other pins which can be connected to the switches, to the push buttons, to the LEDs or to the 7 segment display from the development board. In our Pong game we used the 4 push buttons, 2 for one paddle and 2 for the other paddle for moving them up and down.

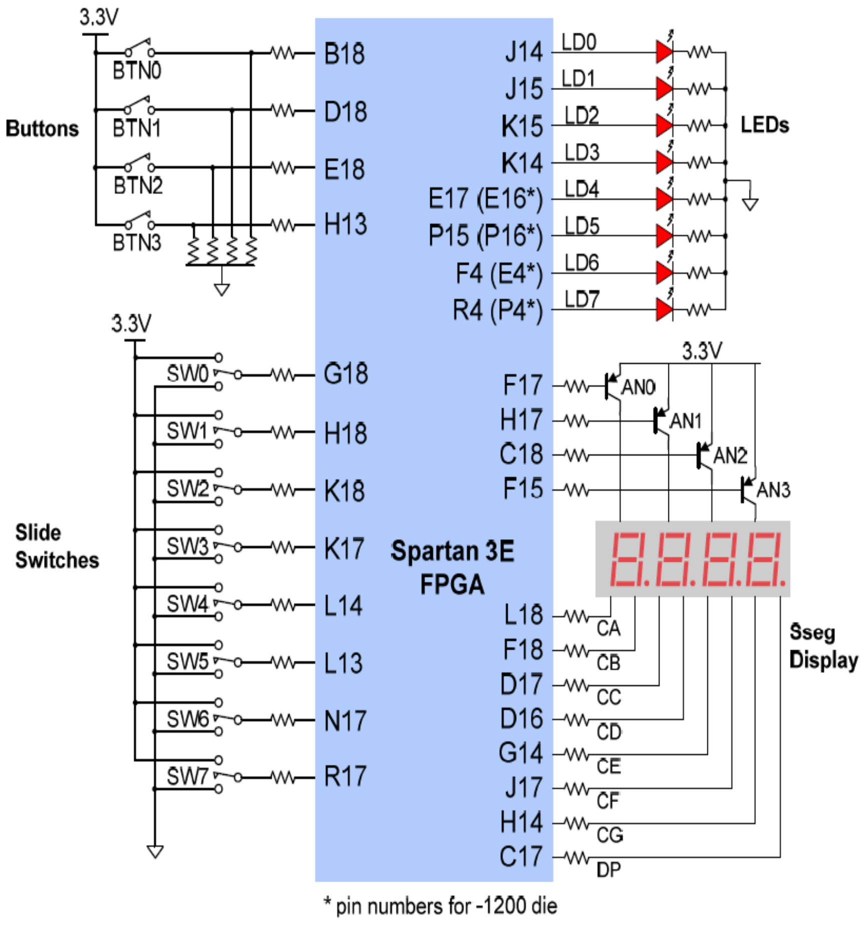


Fig. 4. Spartan-3E pins.

The VGA port on the NEXYS 2 development board is interesting. On Fig. 5 we can see the structure of the VGA port which can be found on the NEXYS 2 development board.

NEXYS 2 board uses 10 FPGA lines to create a port with 8-bit color VGA and two standard lines of synchronization (HS – horizontal sync, VS – vertical sync). Color signals use a resistive divider circuit which together with the 75 Ω termination VGA display, create eight levels of signals, red and green lines, and four signal levels on the blue line.

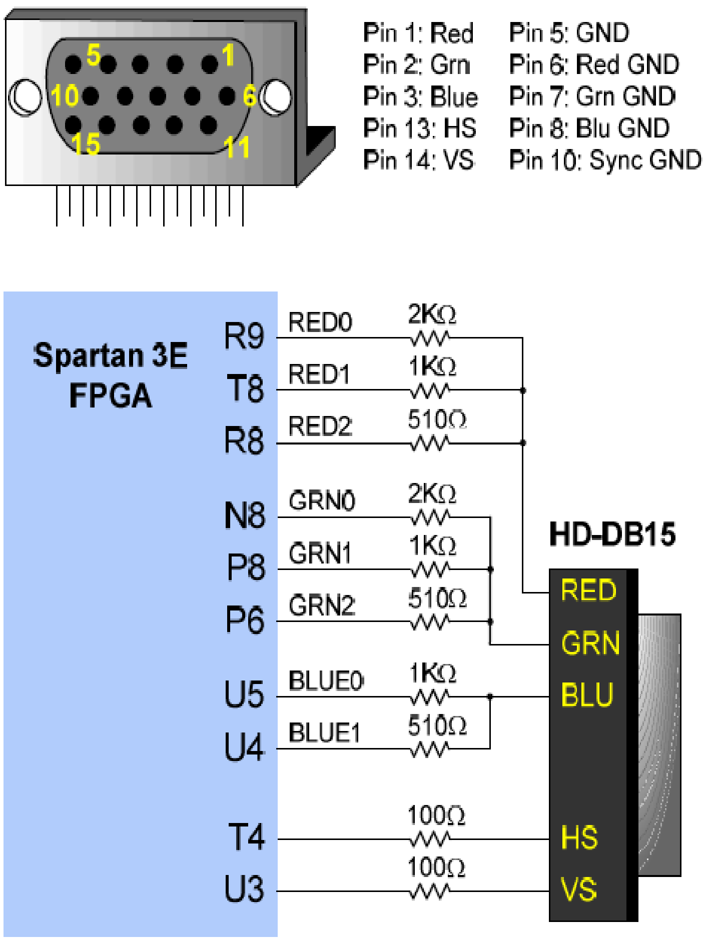
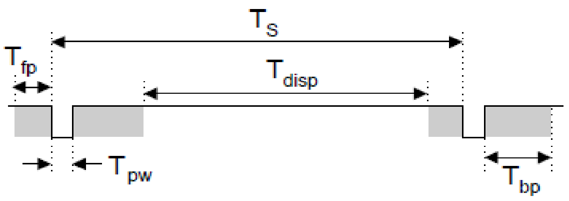


Fig. 5. The VGA port on the NEXYS 2 development board

The VGA timing needs to be set correctly. A VGA controller circuit must generate vertical sync signals – VS and horizontal sync signals – HS and a coordinate delivery of video data on a pixel clock. The pixel clock defines the time available to display one pixel of information. VS signal defines the frequency of the refresh of the screen and is common to all the information on the screen when is redrawn. The minimum refresh frequency is a function of the intensity of the phosphor screen and electronic spot. Basically refresh frequency is in the range of 50-120 Hz. For a screen of 480 lines with 640 pixels per line, using a 25 MHz pixel clock and a refresh of 60 +/– 1Hz, signal timings are shown in Fig. 6. Synchronization time pulse width for intervals of "front" and "back porch" (these intervals are times pre-and post-synchronization, during when information cannot be displayed) are based on observations taken from actual VGA monitors.



Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

Fig. 6. The timing for a resolution of 640x480.

A VGA controller circuit (Fig. 7) decodes the output horizontal sync counter, which is driven by the pixel clock, to generate horizontal sync HS times. This counter can be used to locate any pixel on a given line. Similarly, the output vertical sync counter that increments with each HS pulse can be used to generate VS vertical synchronization times and this number can be used to locate any given line. These two counters (continuously operating) can be used to address a memory.

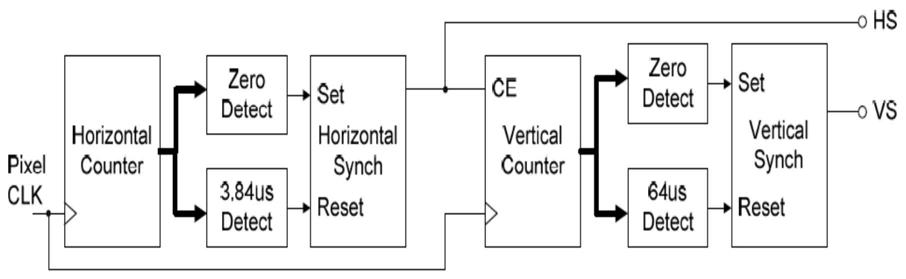
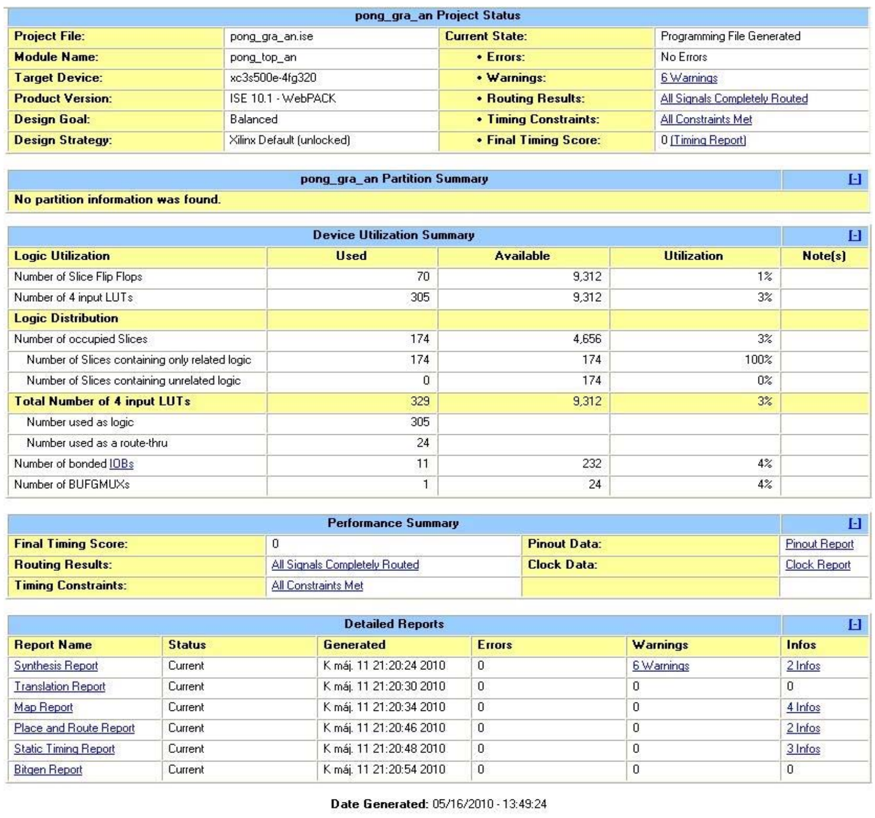


Fig. 7. The circuit for a VGA controller.

3.2. *Implementation of the Pong Game*

On Table 1. We can see the Xilinx ISE implementation summary. Here we have more information about our VHDL code, errors, warnings, used logic structures and the FPGA usage percentage.

Table 1. Implementation summary in Xilinx ISE.



pong_gra_an Project Status			
Project File:	pong_gra_an.ise	**Current State:**	Programming File Generated
Module Name:	pong_top_an	**• Errors:**	No Errors
Target Device:	xc3s500e-4fg320	**• Warnings:**	6 Warnings
Product Version:	ISE 10.1 - WebPACK	**• Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	**• Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	0 (Timing Report)

pong_gra_an Partition Summary	[-]
No partition information was found.	

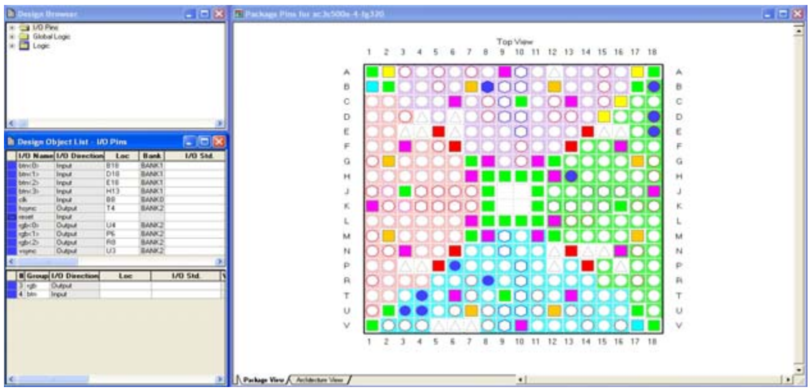
Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	70	9,312	1%	
Number of 4 input LUTs	305	9,312	3%	
Logic Distribution				
Number of occupied Slices	174	4,656	3%	
Number of Slices containing only related logic	174	174	100%	
Number of Slices containing unrelated logic	0	174	0%	
Total Number of 4 input LUTs	329	9,312	3%	
Number used as logic	305			
Number used as a route-thru	24			
Number of bonded IOBs	11	232	4%	
Number of BUFGMUXs	1	24	4%	

Performance Summary			[-]
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	K máj. 11 21:20:24 2010	0	6 Warnings	2 Infos
Translation Report	Current	K máj. 11 21:20:30 2010	0	0	0
Map Report	Current	K máj. 11 21:20:34 2010	0	0	4 Infos
Place and Route Report	Current	K máj. 11 21:20:46 2010	0	0	2 Infos
Static Timing Report	Current	K máj. 11 21:20:48 2010	0	0	3 Infos
Bitgen Report	Current	K máj. 11 21:20:54 2010	0	0	0

Date Generated: 05/16/2010 - 13:49:24

On Fig. 8 we can see the used pins and where are they connected on the FPGA. To know which pins to use we need to read the NEXYS 2 documentation and we need to study Fig. 4. and Fig. 5.



Design Object List - I/O Pins

I/O Name	I/O Direction	Loc	Bank	I/O Std.
btn<0>	Input	B18	BANK1	
btn<1>	Input	D18	BANK1	
btn<2>	Input	E18	BANK1	
btn<3>	Input	H13	BANK1	
clk	Input	B8	BANK0	
hsync	Output	T4	BANK2	
reset	Input			
rgb<0>	Output	U4	BANK2	
rgb<1>	Output	P6	BANK2	
rgb<2>	Output	R8	BANK2	
vsync	Output	U3	BANK2	

B	Group	I/O Direction	Loc	I/O Std.
3	rgb	Output		
4	btn	Input		

Fig. 8. FPGA pin allocation using the Xilinx PACE software.

The *.ucf file is used for allocation of the pins. This file can be created manually or with the Xilinx PACE software. The content of the *.ucf file is presented next.

```
NET "clk" LOC = "B8";

NET "hsync" LOC = "T4";
NET "vsync" LOC = "U3";

NET "rgb<0>" LOC = "U4";
NET "rgb<1>" LOC = "P6";
NET "rgb<2>" LOC = "R8";

NET "btn<0>" LOC = "B18";
NET "btn<1>" LOC = "D18";
NET "btn<2>" LOC = "E18";
NET "btn<3>" LOC = "H13";
```

The Circuit of the Game is not very simple. The chip of the Pong game can be seen on Fig. 9.

The inside structure of the Pong game can be seen on Fig. 10. 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 balls and paddles. We made even the ball to be round by loading the binary values from ROM memory. The last chip is a D-latch for creating a delay for timing.

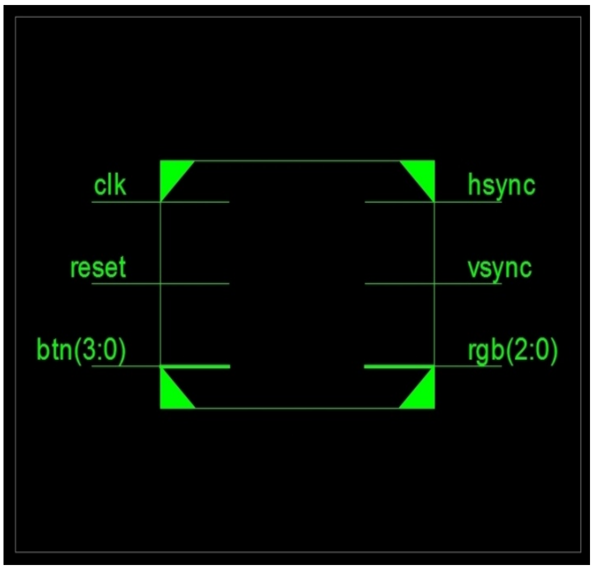


Fig. 9. The Pong game chip.

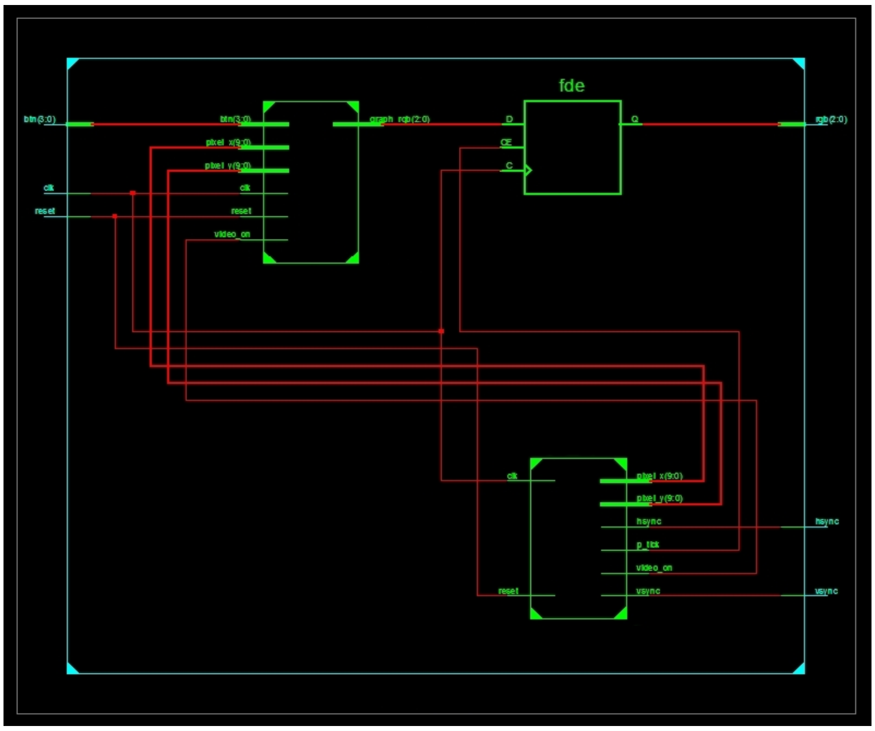


Fig. 10. The inside structure of the Pong game chip.

Explanation of the game rules will be presented next. On Fig. 11 we can see the result of the Pong game on a CRT monitor with VGA input. The paddles are controlled with the push buttons from the NEXYS 2 development board.

The game consists of two blades and a ball. Background color of the paddles and the ball is configurable in software. The background is white, the paddles are red and blue, and the ball is green.

Two players can play the game with the 4 push buttons on the board. If the board is oriented in normal position, with the 4 push buttons right below, the first player, with blue paddle, will have the first 2 push buttons (BTN0 movement up, BTN1 movement down) and the second player, with red paddle, will have the last 2 push buttons (BTN2 movement up, BTN3 movement down).

The ball moves automatically hitting the top and bottom edge of the screen and the paddles when the ball it's hit by a player. If a player misses the ball, than he gets a goal, so the ball goes off the screen to the right or left side. For example, if a player misses the ball on the right, then left player gave a goal to the right player. After this it will be a new ball on the left, so the left player serves. The ball will move straight from left to right as long as the right player manages to hit the ball. If the right player scores a goal, then the ball starts its automatic movement from the right. The game is repeated until it it's stopped. The player, who manages to give the most goals, wins.

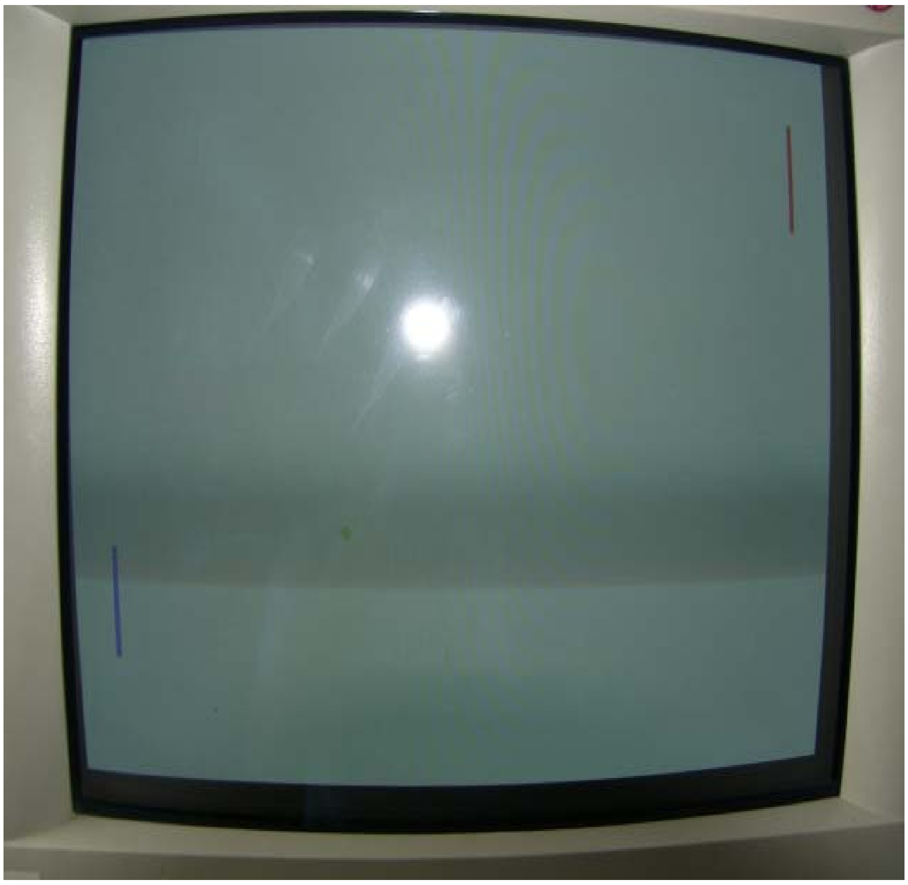


Fig. 11. The result of the Pong game on a CRT monitor.

4. Conclusion

As we could see we created a Pong game on FPGA. Our goal was to reproduce one of the most entertaining games of the computer game history. The Pong game was one of the first electronic games and maybe made the bases of the computer games history.

It was 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 push buttons for controls. Making it on FPGA was a good idea also, because this way we had an embedded system, this way we don't have other dependencies not even operating sys-tem 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, on hardware; this means that we have a standalone device with no speed issues, like if it would be made in software on a microcontroller.

We also plan to choose a capsule for the die with the Pong game.

In future we plan to extend the controls from push buttons to mouse and keyboard on PS/2 interface. We plan also to port the project 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.

References

Aibe, Noriyuki, Yasunaga, Moritoshi (2004): Reconfigurable I/O interface for mobile equipments. IEEE International Conference on Field-Programmable Technology, pp. 359–362.

Bhatt, Deep Vardhan, Toit, D. Du, Hancke, Gerhard P. (2012): Design of a controller for a universal input/output port. IEEE International Instrumentation and Measurement Technology Conference, pp. 647–652.

Kai, Liu, Yuliang, Yang, Yanlin, Zhu (2012): Tetris game design based on the FPGA. 2nd International Conference on Consumer Electronics, Communications and Networks, pp. 2925–2928.

Lim, Kyu-Sam Sam, *et al*. (2008): Design an infrared wireless optical mouse system and a dual-band infrared receiver. 15th IEEE International Conference on Electronics, Circuits and Systems, pp. 810–813.

Pokharel, Punj, Bhatta, Binod, Darji, Anand D. (2011): Optimized drivers for PS/2 and VGA using HDL. IEEE International Conference on Computer Science and Automation Engineering, **3**, pp. 262–266.

Szabó, R. (2014): Creation of the Chips Placement Game with Backtracking Method in Borland Pascal. International Symposium on Electronics and Telecommunications, Eleventh Edition, pp. 85–88.

Szabó, R., Gontean, A., Lie I. (2012): “Smart Commanding of a Robotic Arm with Mouse and a Hexapod with Microcontroller”, 18th International Conference on Soft Computing, pp. 476–481.

Szabó, R., Gontean, A. (2013a): Creating a Programming Language for the AL5 Type Robotic Arms. 36th International Conference on Telecommunications and Signal Processing, pp. 62–65.

Szabó, R., Gontean, A. (2013b): Creating a Serial Driver Chip for Commanding Robotic Arms. Federated Conference on Computer Science and Information Systems, pp. 671–674.

Szabó, R., Gontean, A. (2014a): Pong Game on FPGA with CRT or LCD Display and Push Button Controls. Federated Conference on Computer Science and Information Systems, pp. 735–740.

Szabó, R., Gontean, A. (2014b): Image Acquisition with Linux on FPGA. 22nd Telecommunications Forum, pp. 1007–1010.

Xien, Ye, Zhiquan, Ye (2009): Implementation of PS/2 Mouse and Keyboard Based on Windows CE. International Forum on Computer Science-Technology and Applications, **3**, pp. 318–321.

Zhang, Guoping, Xie, Man-de (2010): Design of visual based-FPGA Ping-Pang game with multi-models. Second Pacific-Asia Conference on Circuits, Communications and System, **2**, pp. 31–34.