The Pong Game
Advanced Embedded Systems

Tobias Lindberg - d04tl@student.lth.se
Carl Wolff - e04cw@student.lth.se
Sajjad Haider - sx06sh0@student.lth.se

October 16, 2008

Abstract

The project introduces the complete design process from problem specification to the design implementation. The project is to design the Pong game in Xilinx Platform Studio. The Pong game can be easily visualized by having a table-tennis game in mind. A two player game where each player will be provided with a side bar (left and right side) controlled by the keyboard and a free moving ball in between the two bars while upper and lower sides are protected by walls.

The score will be displayed on the on-board 7-segment display and the first player to reach five points wins.

Contents

1 Introduction **2**

2 Hardware implementation **3**
2.1 VGA Controller . . . 3
2.2 Color Controller . . . 3
2.3 7-segment Display Controller . . . 4
2.4 PS/2 . . . 5

3 Software **5**
3.1 Pong game . . . 5
3.2 Color Controller . . . 6
3.3 7-segment Display Controller . . . 6
3.4 PS/2 keyboard . . . 7

4 Occupancy of hardware **7**
4.1 Design Summary Report . . . 7
4.2 Timing summary . . . 7

5 User manual **8**
5.1 Hardware . . . 8
5.2 Software . . . 8

6 Problems **9**
6.1 Hardware . . . 9
6.2 Software . . . 9

7 Knowledge **9**

8 Contributions **9**

1 Introduction

The primary goal of this project is to learn how to go from a specification or algorithm to the implementation, how to merge smaller units into a bigger one. The scope of this project deals with designing a pong game. Experience gained during the project is very useful and can be used for developing more complex designs, which is the aim of the course.

There is one major difference from the initial architecture proposal to the final design, mainly the way to send object coordinates to hardware. The intention was to use on-chip RAM (BRAM) to transfer coordinates from software to hardware. This would be a very dynamic way to communicate and extend, but due to problems, we decided to use registers.

This final design is shown in figure 1.

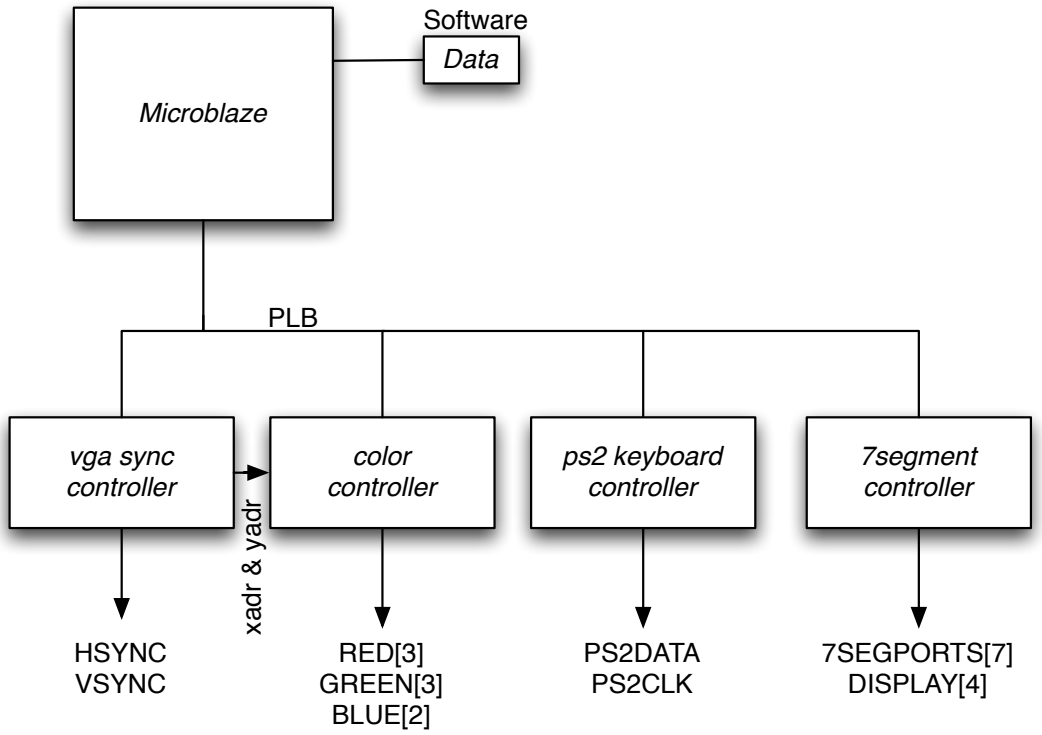

Figure 1: Final design of the architecture

2 Hardware implementation

2.1 VGA Controller

The VGA-controller is responsible for the sync signals sent to the monitor, and by doing this, we also generate addresses for pixel values.

The generated addresses will be divided into x- and y-coordinates with the maximum value of 640x480. These coordinates is connected directly to the color controller. See figure 2

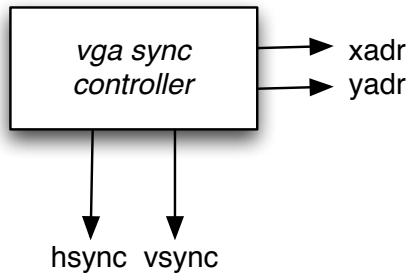

Figure 2: VGA controller with it's connections

2.2 Color Controller

The color controller, figure 3 includes logic to use registers, assigned from software, as limits for different objects. That is, for a ball theres two coordinates specified, x and y. Since all object has been defined as sprites in resolution 16x16, resulting in a array of 255 values, the offset had to be calculated to get the correct pixel drawn.

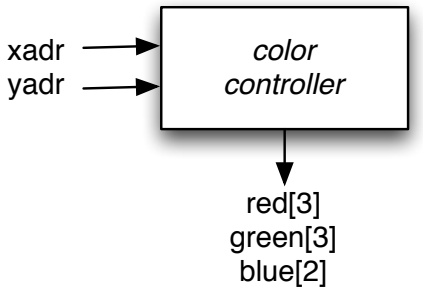

Figure 3: Color controller with it's connections

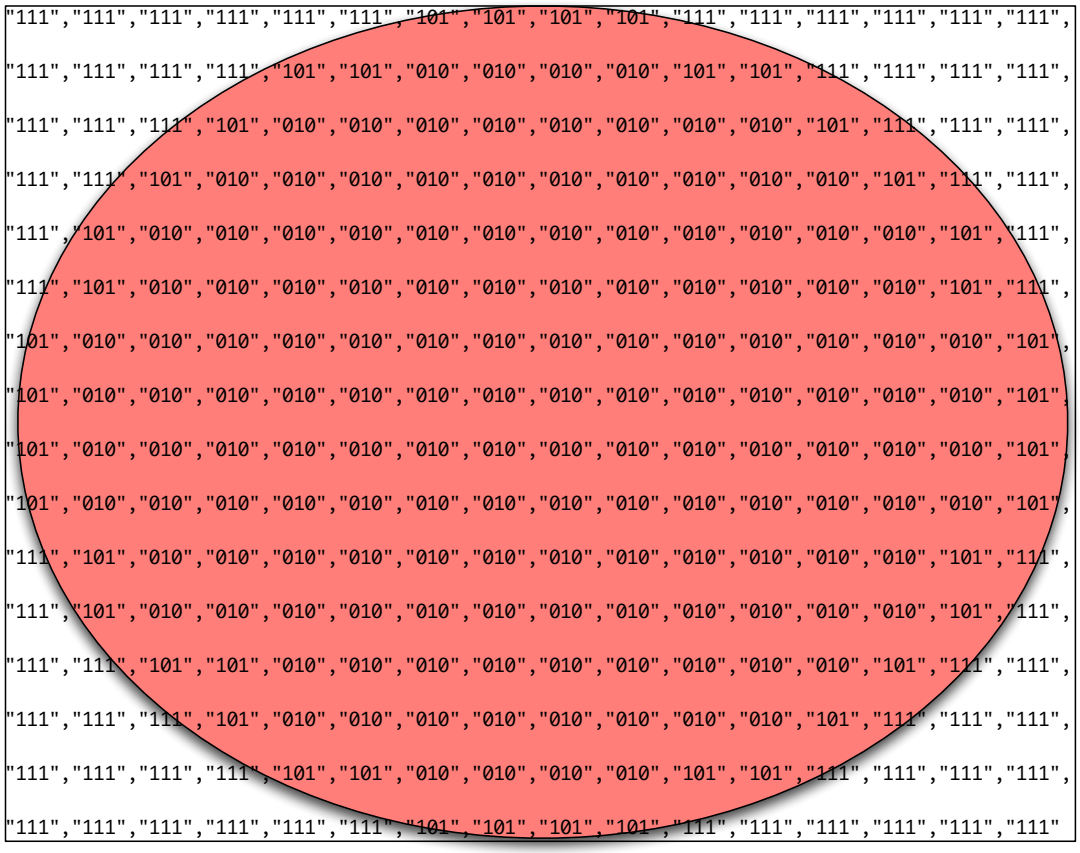

Figure 4: The ball sprite, 16x16 values

2.3 7-segment Display Controller

When the software needs to reset or update the players score, it will call a method, which will write the values in the register for the 7-segment. Due to that only one register is used for the entire display, the score has to be concatenated into a 32 bit integer before saving it to the register.

The 7-segment controller on every clock tick read its register, moves the players score to its corresponding LED and finally decodes the score in accordance to the 7-segment LED.

The 7-Segment basically has three components; the first component divides the data and assigns the players score to the correct digit place. The multiplexer, in every clock cycle, enable the digits and send its corresponding score value to another component, which generates the correct signals for the 7-segment LED display. It is important to mention that since only the first and the fourth LEDs will be used to display the scores

for two players, then the second and the third LEDs should be turned off. This can be easily achieved by sending all '1' to the related LED.

The controller is directly connected to the display pins as shown in figure 5.

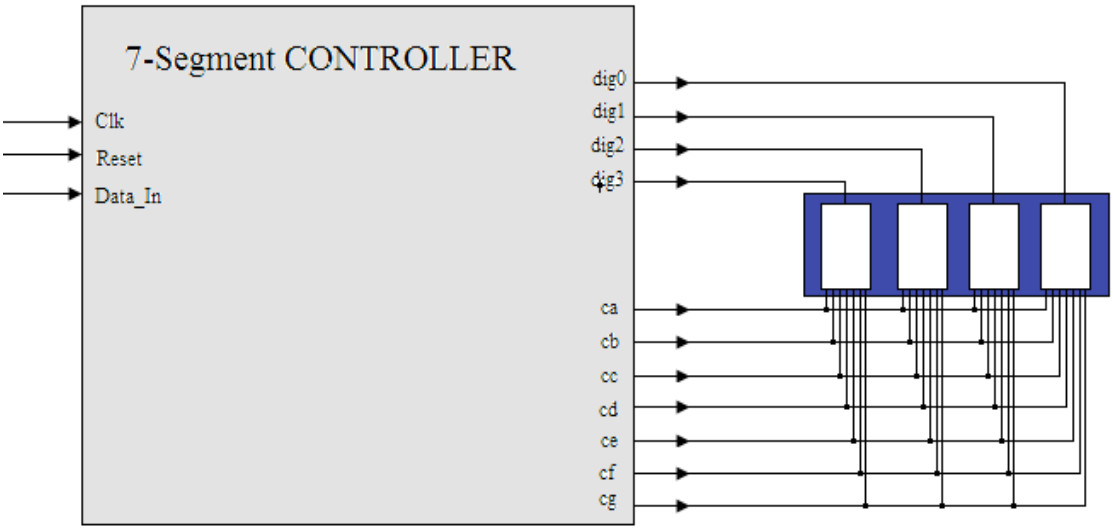

Figure 5: Color controller with it's connections

2.4 PS/2

The XPS core has been used to provide keyboard input using a interrupt handler.

3 Software

3.1 Pong game

All calculations for the movement of the balls and the pads is done in the software. Basically the whole game is written in software and the hardware part is only for displaying the results of all the calculations. The communication between hardware and software is done through registers with the help of the physical addresses of the hardware.

```
1 void pong_game(void)
  {
    /* Initializes the keyboard */
    init_key();

    /* Initializes the colors used */
    init_color_table();
```

```
    /* place out the walls */
    set_walls_pos(WALL1_Y, WALL2_Y, WALL_SIZE);

    /* Initializes the game */
    init_game();

    while(1)
    {
      if(play){
        /*Move the ball*/
        move_ball();

        /* delay */
        int i;
        for (i = 0; i < 600;i++);
      }

      key_respons();

      if( p1_score == 5 || p2_score == 5)
      play = FALSE;
    }
}
```

Listing 1: Pong game main loop

3.2 Color Controller

Defines the methods used to comunicate with the color controller in hardware. It contains methods that sends the coordinates of the ball and the pads, the size of the pads in height and the walls and it's corresponding size in height.

Colors is mapped through two registers that contains the colors chosen in the sprite definition.

```
void update_ball_pos(int x, int y);
void update_pads_y_pos(int y1, int y2);
void init_pads_x_pos(int x1, int x2);
void set_walls_pos(int y1, int y2, int size);
void set_bar_size(int size);
void init_color_table();
```

Listing 2: Color controller header file

3.3 7-segment Display Controller

Defines the method used to send the present score of the game to the display on the Digilent board.

```
void write_score(int player1, int player2);
```

Listing 3: 7-segment display controller header file

3.4 PS/2 keyboard

Defines the method used to receive the keys pressed on the keyboard. The keys are received in a vector of length 10 where sections of code in the pong game is woken up if certain keys are pressed.

```
void init_key();
void get_key(char keys[10]);
void exit_key();
```

Listing 4: Keyboard header file

4 Occupancy of hardware

4.1 Design Summary Report

Hardware	Amount(#)	Used percentage(%)
External IOBs	107	42
External Input IBUFs	14	100
External Output IOBs	63	84
External Bidir IOBs	18	100
BSCANs	1	100
BUFGMUXs	2	8
DCMs	1	12
MULT18X18SIOs	3	10
RAMB16s	16	57
Slices	5895	67
SLICEMs	264	6

4.2 Timing summary

Minimum period: 19.976ns
Maximum net skew: 1.966ns

Estimated clock frequency: 50 MHz

5 User manual

5.1 Hardware

Make sure that the hardware, the Digilent Nexys-2 Spartan-3E-1200, is connected like shown in the figure 6.

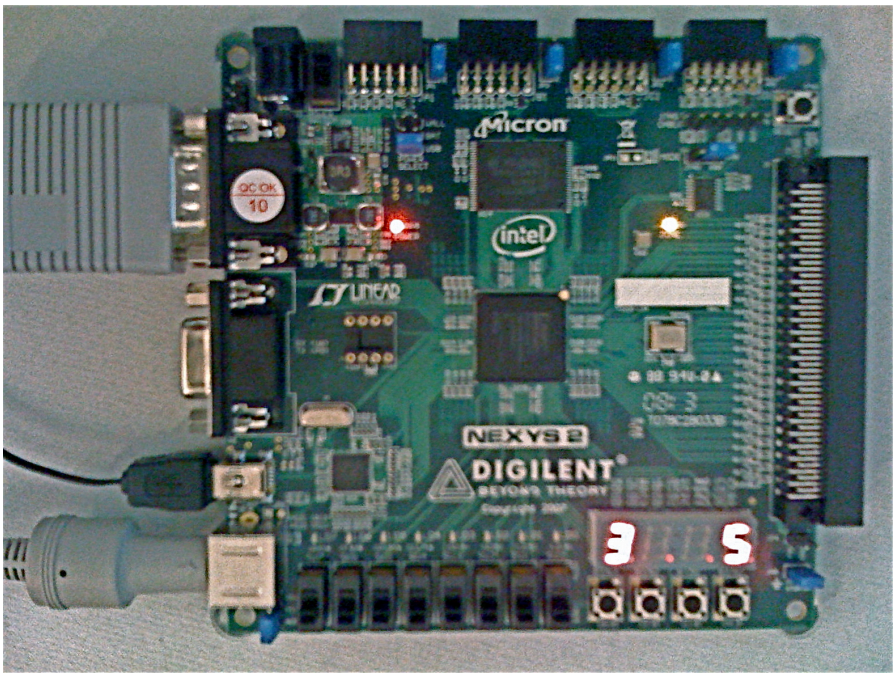

Figure 6: Digilent Nexys-2 1200 board, with VGA, USB (power+data) and PS/2 connected.

If the XCF04S ROM hasn't been reprogrammed, Digilent self-test will be started on power up. This is normal.

5.2 Software

The project is only compatible with Xilinx Platform Studio (XPS) version 10.1 or above. If you run into problems during start up, please download all updates for your application, or make necessary changes by manual editing of erroneous source file.

Open the project in Xilinx Platform Studio, choose File-Open Project..., then open up the system.xmp project file in the root of the projects path. Using XPS you can change variables as needed to fit your board. When you're done, compile it using menu item: 'Device configuration'-'Update bitfile'.

To send the project to the board, use Digilent ExPort. There is a bit file available in:

'...projectpath/implementation/download.bit'.

6 Problems

6.1 Hardware

The main problems came with the software Xilinx EDK. Since it's not totally intuitive, we had problems doing basic stuff such as connecting signals from one core to a actual pin on the board. Some problems also occurred during the installation and programming since Windows wasn't always our choice of operating systems at our computers at home.

We also had some problems getting the color control to work. A variable assignment proved to be incorrect which made the color mapping invalid.

6.2 Software

Since the game is quite easy there were no big problems. We had one "problem", the speed of the game. This had to be slowed down to actually see the ball. We took the easy way and used a single 'for-loop' which pauses the software by increasing a number.

7 Knowledge

The clear picture of the architecture was done in an early stage but how EDK works was a different issue.

If we would have the opportunity to start over, then we would probably have tested all parts in ISE before using them in EDK. To synthize is time consuming in EDK because of the Microblaze and other parts not concerning the specific component you want to test.

8 Contributions

The software solution for the Pong game is written entirely by Tobias Lindberg (TL), with influences from existing Pong games found on the Internet. The initial software was based on a found solution, just to have a hardware base, but has been replaced, leaving nothing but the name from the initial software. All modifications has been written by Lindberg. All testing needed was also done by Lindberg.

The project included three group written cores, the VGA-, color and the 7-segment-IP. The cores was essential to make this project work, and we divided the work equally from

the beginning.

Initial architecture was done by Tobias Lindberg, extended architecture by Carl Wolff and Lindberg after some discussion with teachers. Final arch. can be seen in figure 1.

The VGA-controller was written by Lindberg. Idea is quite general part, but some suggestions was made by Wolff. Lindberg ran into problems in the last stage, testing. The problems that occurred had to do with the front porch and the back porch and Linberg decided to leave his solutions and made a modified solution of the one on the course homepage. A nother problem that became a big issue was the fact that when the VGA controller was tested we sent colors to the screen all the time not caring about if we where in the visible area or not wich made the howl screen black, this error got found with the help of a members from another project, Lim, Wee Guan.

The Color-controller was written by Wolff with and co-op by Lindberg because of the need for address generators. He ran into problems during sprite usage, but solved that quite easily by modifying limits (thanks Lindberg for pointing this out). Lindberg assisted with Pong software changes as needed. The IP provided to the project came with a software header file that had all definitions about how to use the color controller. Some discussion was made with Carl Hakenas (FPGA Hero) about how to do it more dynamic.

The 7-segment controller was done by Sajjad Haider. Since another group also worked on the 7-segment display component, development has been done in conjunction with Imran Khan. Since the component requirements differs, all modifications has been made by Haider. Software needed to communicate with the hardware was developed by Haider by suggestions of Wolff.

The PS/2-software part was done by Wolff. Problems during this initial stage, from application - XPS - point of view, was solved by help from teachers and Adolfo de la Calle and Puneet Acharya.

Report- and presentation draft, was written by Haider, and the main text part together by Lindberg and Wolff. Final merging and graphics was made by Wolff with assistance of the group members.

Finally we would sincerely like to thank both supervisors for their suggestions during entire project. We would also like to thank the other groups for sharing their knowledge and discussion during the project.