Implementation of Ping Pong on FPGA using HDMI screen

CS 288. Project Report: (*Group 15*)

Rohan Das	Aamod Kore	Shital Godara	Anirudh Vemula	Avneesh Kumar
110050001	110050004	110050014	110050055	110050031

April 14, 2013

1 Introduction

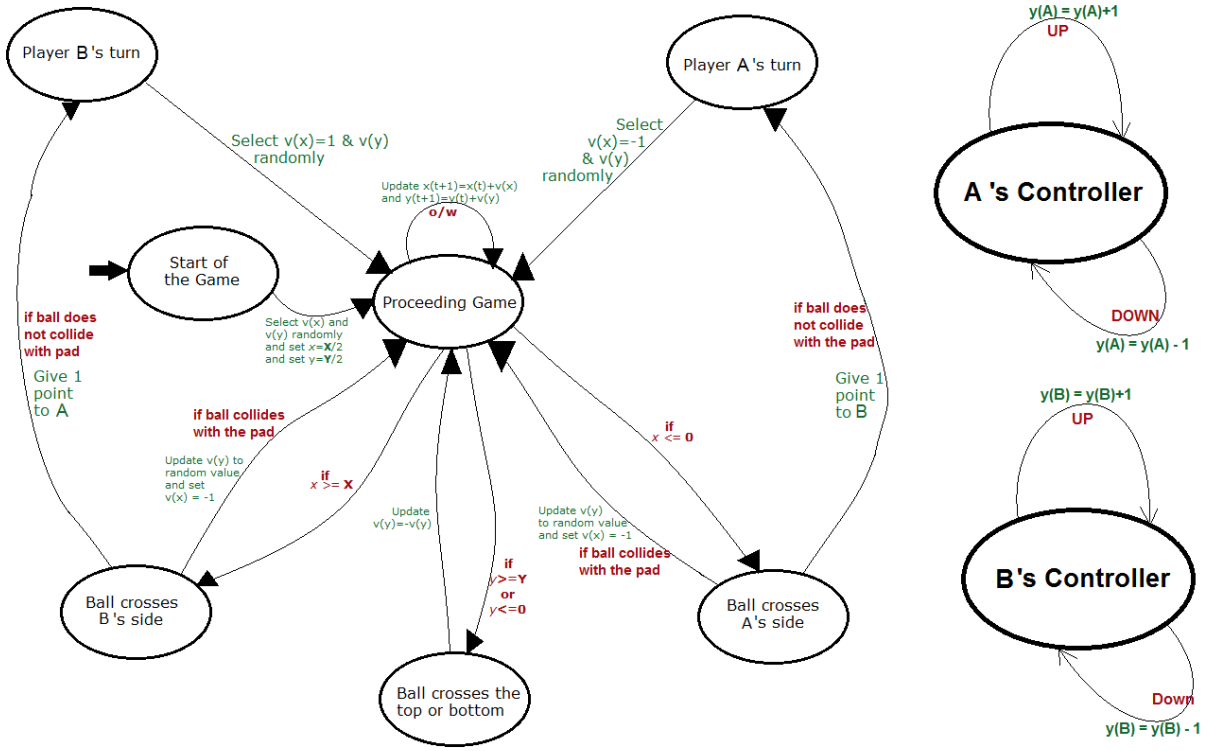
We have made implemented the basic 2-player ping pong game on a Spartan-6 FPGA and using an HDMI screen for output. This report contains the design of the logic and the documentation of the code of our final project. We have made our design rst in VHDL, tested it using a test-bench in the Xilinx ISE and then implemented it on the FPGA. We used the internal clock to drive the entire system. The 8 on-board LEDs are used to display the scores of the two players in binary form. The scores are also displayed on the HDMI screen . We will use push buttons as the control switches for the players and the first 4 DIP switches to control the type of output. The las two DIP are used for pause/start for the game and for the display on/off switch.

2 Project Design

2.1 Design Overview

- We declare 5 variables, 1 for each paddle and 3 for the ball. y(A) and y(B) contain the hieghts of paddle A and B respectively. x, y stores the position of the ball along x-cordinate, y-cordinate respectively and v(y) stores the velocity along y-cordinate. Note that v(x) doesnt change during the process so we keep it a constant. Scores of A and B will also be maintained by some variables.
- We will need 5 input buttons from the FPGA board: 4 to move the 2 paddles UP and DOWN. The 5 button starts the game
- We find the state of the system at each clock edge. The ball moves horizontally and vertically by a distance of v(x) and v(y) respectively. If a player pushes the UP or DOWN button, his ycordinate changes by +10 or -10 respectively.
- We assume all collisions to be elastic. When the ball hits a paddle, we change v(y) to a number depending on where the ball hits the paddle.
- The length of the board will be an integral multiple of v(x). This ensures that our condition for collision of the ball with a paddle, shown in the state diagram, is valid/correct.

2.2 State Diagram



3 IO Specifications

The code has to be synthesised and implemented on a Xilinx Spartan-6 FPGA. The display ouput will be given on the HDMI OUT port. The scores are also output on LEDs LD0 to LD2, and LD4 to LD6.

As for the inputs the DIP switches SW0 to SW3 control the type of display output(i.e. resolution and clock speed) and needs to be set appropriately according to the screen. The best result is when all 4 switches are low (off). The switch SW7 is used for Pause/Play in the game and the switch SW6 is used for ON/OFF for the display output. The player controls are the 4 push-buttons BTNL, BTNR, BTNU and BTND. The RESET button on the FPGA serves as the Game Reset (to start a fresh new game).

4 Module/Entity Design in Code

We have the following entities declared in our project :

- ball
- slow clocking
- game
- player
- pong block

Now we describe our logic/design of our implementation in the subsequent sections:

4.1 ball

ball.vhd contains the implementation of the ball entity. It contains the following inputs and outputs:

- reset : in STD-LOGIC : When this is set to 1 the game starts off from the begining.
- new-clk : in STD-LOGIC : Clock for this module. On an positive edge trigger, the game moves.
- pos-A : in STD-LOGIC-VECTOR (8 downto 0) : y-cordinate of the pad A.
- pos-B : in STD-LOGIC-VECTOR (8 downto 0) : y-cordinate of the pad A.
- pos-x : out STD-LOGIC-VECTOR (9 downto 0) : x-cordinate of the ball.
- pos-y : out STD-LOGIC-VECTOR (9 downto 0) : y-cordinate of the ball.
- scr-A : out STD-LOGIC-VECTOR (3 downto 0) : Score of player A.
- scr-B : out STD-LOGIC-VECTOR (3 downto 0) : Score of player A.

The main process of this ball can be divided into 3 parts, as shown in the commenting of the code:
Note: Here we have considered the screen to be 640x400 pixels. Also, we have taken the horizontal velocity of the ball to be 20 pixels per clock edge and the initial vertical velocity to be 16.

- **Updating ball's x coordinate:** If the ball reaches the horizontal ends of the screen i.e. if x=0 or x=640, the ball checks whether it hits the respective pad or not. If not, then scores are updated and the game is started again. If it has hit, then the velocities are changed considering elastic collision.
- **Updating ball's y coordinate:** Here, we check whether the ball i.e. if x=0 or x=640, the ball checks whether it hits the respective pad or not. If not, then scores are updated and the game is started again. If it has hit, then the velocities are changed considering elastic collision.

Note: The player whose reaches a score of 7, wins the game and the game window closes. Also, the y-velocity of the ball on hitting the pad is decided by the position it hits the pad. It is equal to the position the ball hits the pad minus the position of the center of the ball. We have also made the ball to wait for 30 slow-clock pulses on a player losing the game.

4.2 slow-clocking

clocking.vhd contains the implementation of the ball entity. It contains the following inputs and outputs:

- sys-clk : in STD-LOGIC : Input clock is the FPGA clock. This is 100 MHz clock.
- out-clk : out STD-LOGIC : The above input clock is convert to a 10 Hz clock is the output.

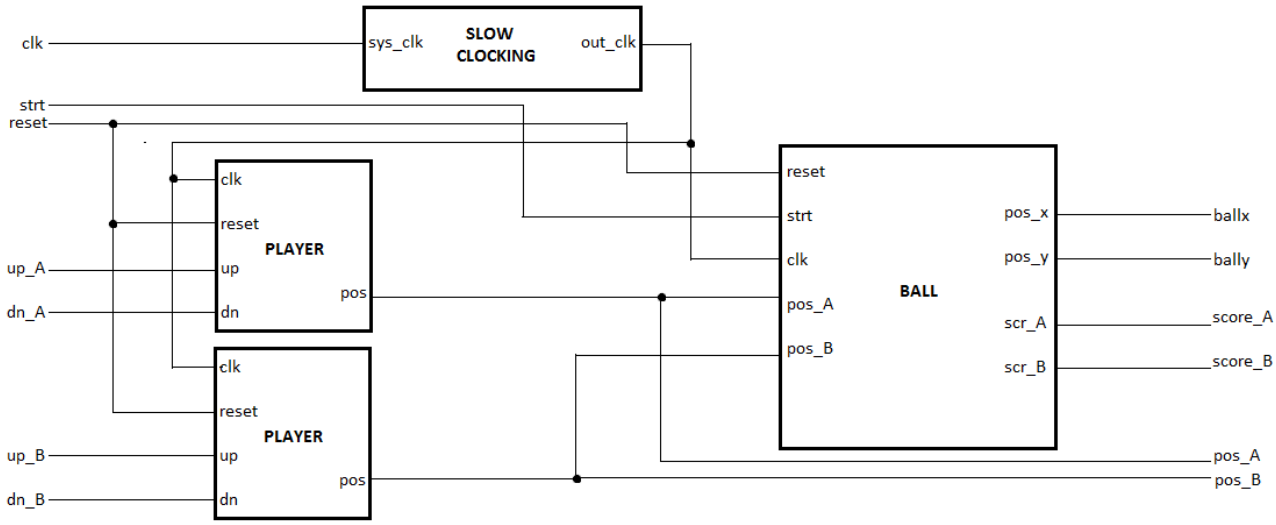
We count positive edge trigger of the FPGA clock till 5000000. After this, if the out-clk is 1, it is set to 0 and 0 to 1. Thus out-clk has 10Hz frequency.

4.3 game

game.vhd contains the implementation of the game entity. It contains the following inputs and outputs:

- reset : in STD-LOGIC;
- strt : in STD-LOGIC;

- clk : in STD-LOGIC;
- up-a : in STD-LOGIC;
- dn-a : in STD-LOGIC;
- up-b : in STD-LOGIC;
- dn-b : in STD-LOGIC;
- pos-A : out STD-LOGIC-VECTOR (11 downto 0);
- pos-B : out STD-LOGIC-VECTOR (11 downto 0);
- score-A : out STD-LOGIC-VECTOR (3 downto 0);
- score-B : out STD-LOGIC-VECTOR (3 downto 0);
- ball-x : out STD-LOGIC-VECTOR (11 downto 0);
- ball-y : out STD-LOGIC-VECTOR (11 downto 0));



The entity ***game***

This is the module where the main game is integrated. Here all the inputs and outputs are mapped to corresponding coordinates accordingly. It contains 2 **player** components, one **slow-clocking** component and one **ball** component and puts them all together. It thus takes the controls(UP and DN for both players as the input and gives the players' scores, the paddles' positions and the ball's position and velocity as the output.

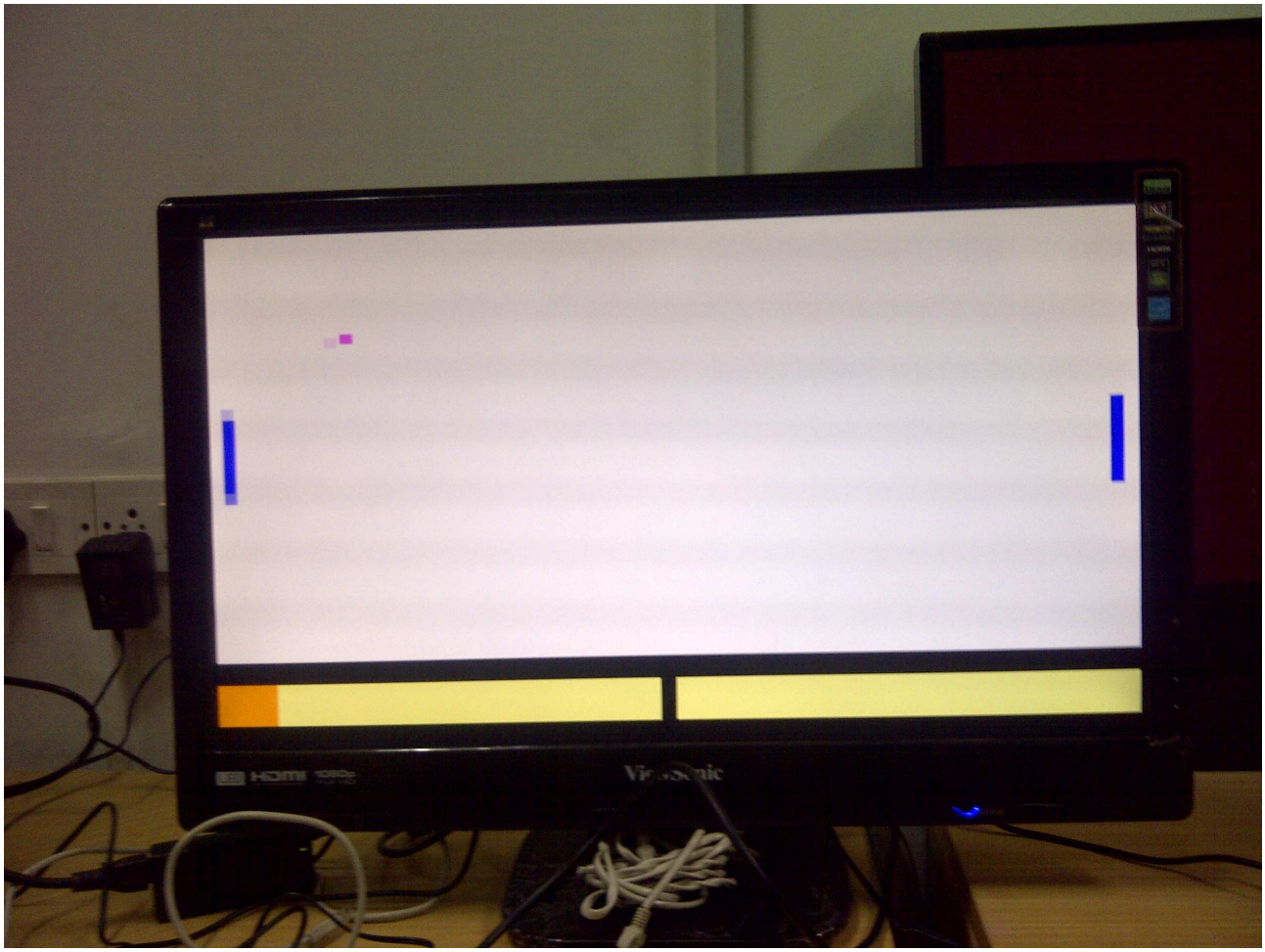
4.4 display

display.vhd contains the implementation of the display entity. It contains the following inputs and outputs:

- reset : in STD-LOGIC;
- ya, yb: in std-logic-vector (11 downto 0);
- px, py: in std-logic-vector (11 downto 0);
- bx, by: in std-logic-vector (11 downto 0);

- sa, sb: in STD-LOGIC-VECTOR(3 downto 0);
- red: out STD-LOGIC-VECTOR(7 downto 0);
- green: out STD-LOGIC-VECTOR(7 downto 0);
- blue: out STD-LOGIC-VECTOR(7 downto 0));

It takes all the outputs of the game, i.e. the position of the pads, ball, scores, x-counter and y-counter, and outputs the RGB values at the particular pixel value. Thus, it is able to control the display of the game. The scores are displayed at the bottom of the game window in the form of rectangular bars with length proportional to the score.
The following photograph, we took while running our code on the FPGA, depicts the display on the HDMI screen.



Demonstration of the implementation on HDMI output

4.5 player

player.vhd contains the implementation of the player entity. It contains the following inputs and outputs:

- up : in STD-LOGIC;
- dn : in STD-LOGIC;
- clk : in STD-LOGIC;

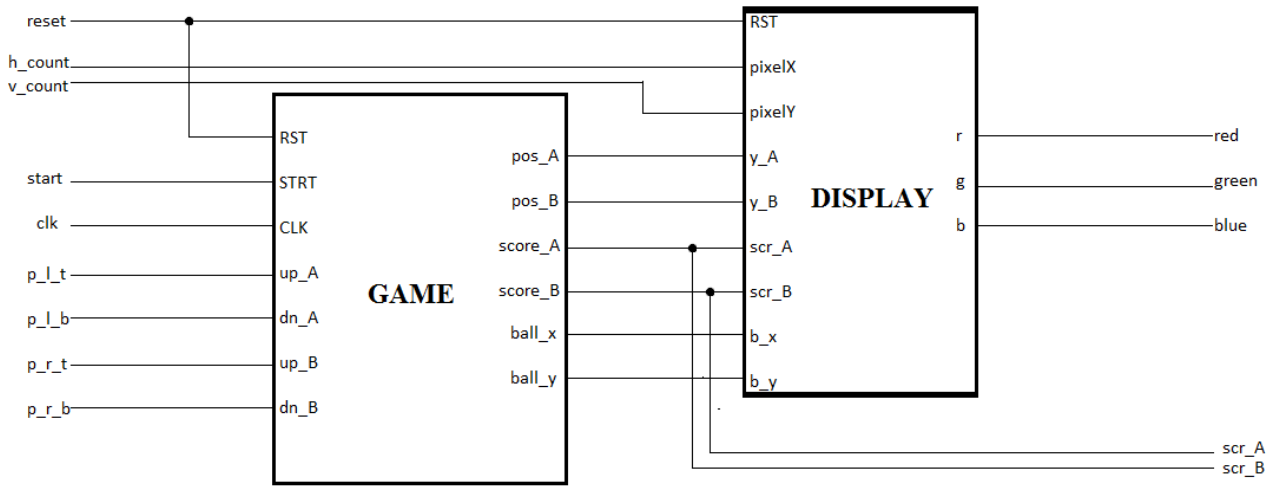
- reset : in STD-LOGIC;
- pos : out STD-LOGIC-VECTOR (11 downto 0));

This entity controls the behavior of a player. Each player has a pad of length 80 and on $up = 1$, it moves up by 10 pixels and when $down = 1$, moves down by 10 pixels.

4.6 pong-block

pong_block.vhd contains the implementation of the ball entity. It contains the following inputs and outputs:

- clk : in STD-LOGIC;
- reset : in STD-LOGIC;
- p-l-t : in std-logic - push button to move left paddle up
- p-l-b : in std-logic - push button to move left paddle down
- p-r-t : in std-logic - push button to move right paddle up
- p-r-b : in std-logic - push button to move right paddle down
- strt : in std-logic - push button for start
- h-count: in std-logic-vector(11 downto 0);
- v-count: in std-logic-vector(11 downto 0);
- scr-a: out std-logic-vector (3 downto 0);
- scr-b: out std-logic-vector (3 downto 0);
- red : out std-logic-vector(7 downto 0);
- green : out std-logic-vector(7 downto 0);
- blue : out std-logic-vector(7 downto 0));



The entity ***pong_block***

This entity integrates the game with the display. Thus it contains all the inputs and outputs of the entire project and maps it to respective components. Besides the controls of the game it also takes as input two 11-bit vectors for the x and y count of the pixel and finally outputs the red, green and blue values at the corresponding pixel in the form of three 7-bit vectors.

4.7 HDMI Output Drivers

The folder **xapp495** contains the code files for the HDMI Output Drivers. These are written in VHDL and Verilog and are taken from an internet reference suggested by the course TAs. These drivers basically help to connect the pong_block component to the HDMI ouput.
The **vtc_demo.v** contains the topmost module that needs to be sythesised and implemented.