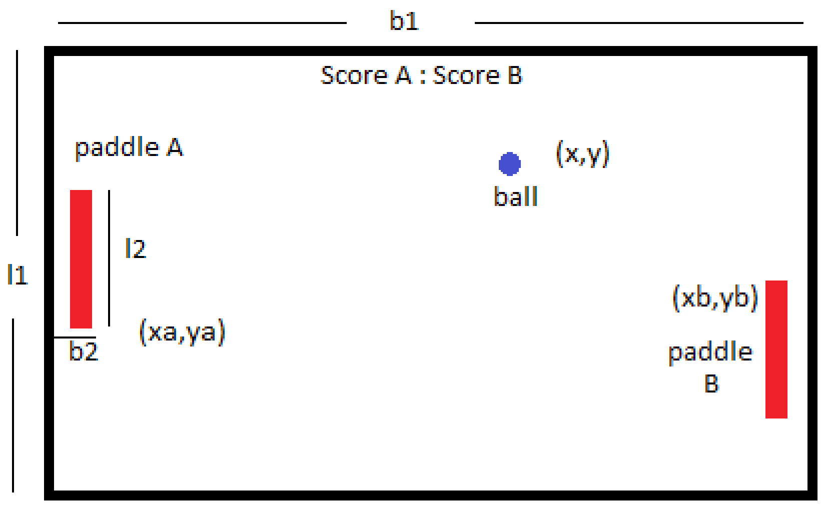
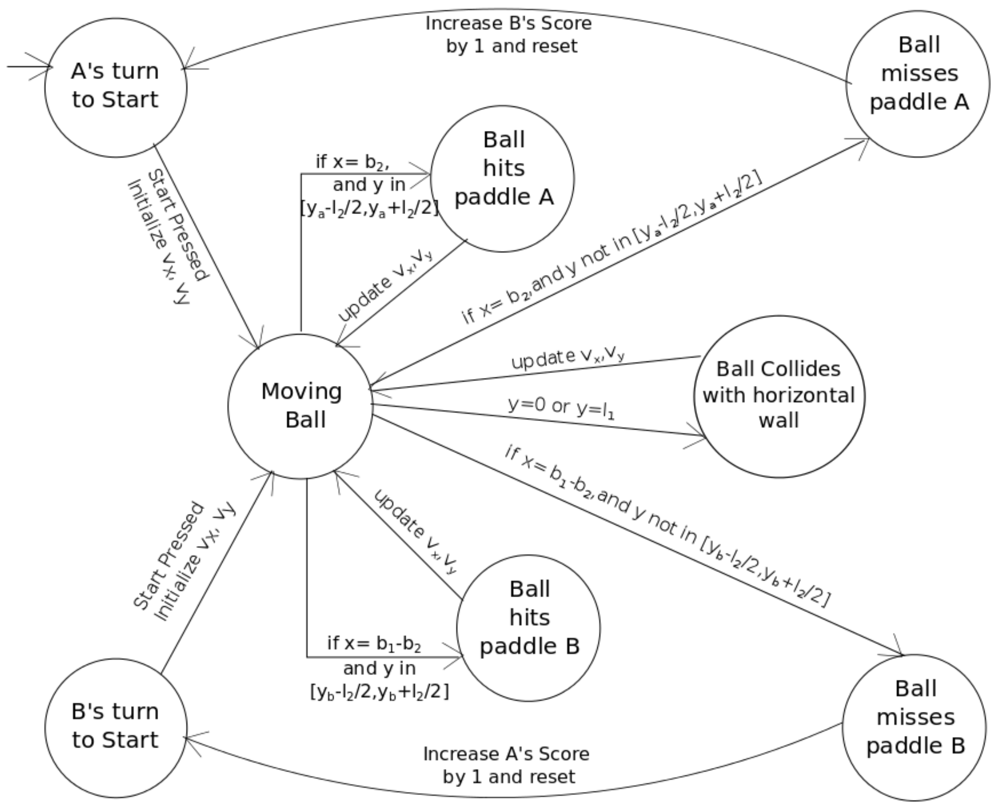
# PING PONG

## Introduction:-

Ping pong is a two-player game played on a screen where there are two paddles for each player and one ball. The paddles are at the two vertical walls of the screen and are constrained to move vertically. The ball keeps moving between the paddles and is reflected upon hitting the paddles or the walls. The task of each player is to prevent the ball from hitting the vertical wall on his side. When one player is unable to do so, then the other player wins a point and the game continues like this.



In this project we have implemented the ping pong game on the SPARTAN6 board and displayed the play on an HDMI screen. The game has been modelled as a state machine with 8 states. The states are changed depending on the position and motion of the paddles/ ball as can be seen in the figure shown below.


A's turn to Start
Increase B's Score by 1 and reset
Ball misses paddle A
Ball hits paddle A
if x= $b_2$, and y in $[y_a-l_2/2,y_a+l_2/2]$
Start Pressed Initialize $v_x$, $v_y$
update $v_x$,$v_y$
if x= $b_2$,and y not in $[y_a-l_2/2,y_a+l_2/2]$
Moving Ball
update $v_x$,$v_y$
Ball Collides with horizontal wall
y=0 or y=$l_1$
if x= $b_1$-$b_2$,and y not in $[y_b-l_2/2,y_b+l_2/2]$
update $v_x$,$v_y$
Start Pressed Initialize $v_x$, $v_y$
Ball hits paddle B
if x= $b_1$-$b_2$ and y in $[y_b-l_2/2,y_b+l_2/2]$
B's turn to Start
Ball misses paddle B
Increase A's Score by 1 and reset

We have used many variables and constants as a part of the game implementation. The variables are ya , yb , x , y , scorea , scoreb ,
denoting the y-coordinates of paddle A & Paddle B, x & y coordinates of the ball and the scores of player A and player B respectively. The constants are ya , yb , x , y , scorea , scoreb , denoting the y-coordinates of paddle A & Paddle B, x & y coordinates of the ball and the scores of player A and player B respectively.

The output is displayed on the HDMI screen through the HDMI out port on the Spartan6 board. Five signals are built-in our code to control the VGA output: horizontal_sync, vertical_sync , red, green and blue.

## Implementation:-

We have used a vtc_demo.v project from http://dejazzer.com/ee478/ lab9 as our hdmi encoder and driver. This project initially drew colored bars across the screen depending upon resolution which can be set from dip switches on the board.

So we just implemented our code logic inside hdclrbar.v where it was setting the colors according to pixel address. So we kept all other code same and added some new buttons and two new modules namely pong_block and scores for our implementation of the code. All other code is mostly the same as downloaded.

In **pong_block** we have :-

**Control Signals:**
**clk** : 75Mhz clock
**reset** : Reset button
**p_l_t** : push button to move left paddle up
**p_l_b** : push button to move left paddle down
**p_r_t** : push button to move right paddle up
**p_r_b** : push button to move right paddle down
**start** : push button for strt

**HDMI Signals:**
**h_counter** :horizontal pixel counter
**v_counter** : vertical pixel counter
**red** : red signal for the current pixel
**green** : green signal for the current pixel
**blue** : blue signal for the current pixel

1. We have process called states to set the state according to input clk and reset. It will set state to “moving” if reset is 0 otherwise it is “stopped”.

2. We have a process to make a slower clock called Move_Clk on which all the movement depends. This is a 75 Hz clock and runs only when state is moving otherwise it is stopped. So our game works only in moving state othewise it remains stopped.

3. Then we have process for paddle_left_move and paddle_right_move. These two processes are responsible

for movement of paddle on the screen. In paddle_left_move if p_l_b is 1 than it moves down untill it reaches maximum similarly if p_l_t is 1 than it moves up untill it reaches top. Similarly for the right paddle. These movements are synchrnous with the clock and occurs when Move_clk positive edge occurs. If at any time reset becomes 1 we reset the position of the paddles to center of the screen.

4. Next we have ball_move process which is responsible for ball movement and score updation. If reset is 1 at any time we reset ball to middle and reinitialize the scores. When restart is 1 we count till 20 Move_clk signals and restart ball from center.
   Otherwise ball is moved according to its x and y velocity. Before moving the ball it is checked if the next position is permissible if not than velocities are updated.
   1. When ball hits up and down wall y velcoity is negated and x remains constant.
   2. When ball is in the range of paddle hitting it hits it and x direction is reversed and y velocity is updated depending on place of collision on the paddle. Which depennds on the sign of the velocity and loction of ball's center w.r.t. to paddle.
   3. If ball misses paddles and goes behind and hits back wall, score is increased accordingly and restart is set to 1 so it waits and than game again start from the center.
5. Next is a draw process which draws on the screen according to horizontal and vertical counters. It paints the scores in 10 X 10 two blocks according to the value of 10 – 10 bit vectors. Also the ball and paddle are drawn according to paddle and ball position at that instant. Otherwise background is white.

In **Scores** we have :-

scr : 3 bit score

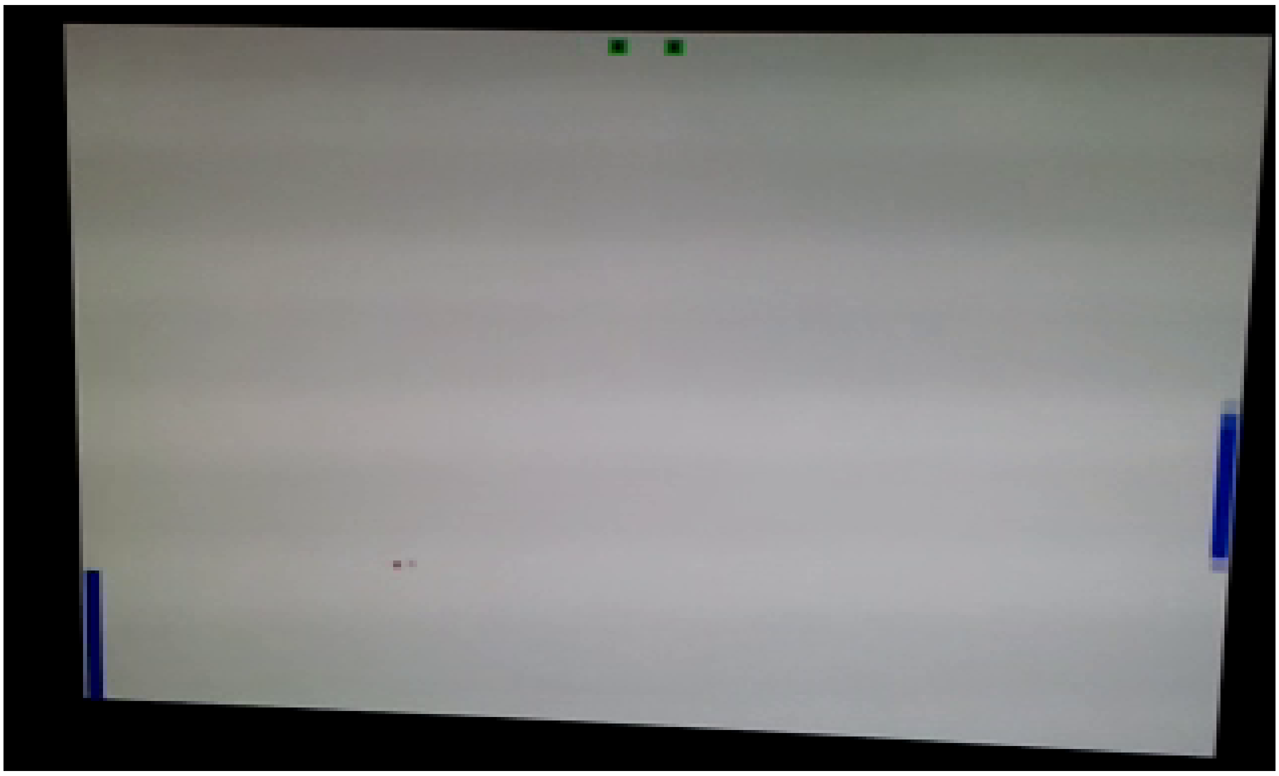
a1-a10 : 10 bit vectors representing digits

In this we have a case statement which according to value of scr assigns a1-a10 value of 10 bit vectors which have 1's and 0's representing digits.

**UCF File :-**
In the ucf file we have mapped 4 push buttons for playing controls and 1 reset button to reset the game any time.

Reset – T5 – 2nd DIP switch from left
left up – P4
left down – P3
right up – N4
right down - F6

## Screenshot :-



Screenshot of the game with blue paddles and red ball

Score at the top of the screen 1-2

Video Link of playing : https://www.youtube.com/watch?v=2MNv5VHrgm0