DIGITAL HARDWARE DESIGN
PROJECT REPORT

TWO DIMENSIONAL
PING PONG

By:
GAGAN BANSAL -- 2003CS10162
KARAN MANGLA -- 2003CS10166

INSTRUCTOR: Prof. M. BALAKRISHNAN

GROUP NO: 2
**Game Specifications:**

We have made a two-dimensional ping pong game. In this game both players have one side of the table (8X8LED display) which represents their goal. The aim of the game is to put the ball through the other player’s goal. Each player has a paddle to hit the ball. When hitting the ball each player will get an option of the amount of spin that he/she wants to impart to the ball. However a greater amount of spin will lead to a slower speed of the ball when hit by the player. The person to first
score 16 points will win the game. Further this game also comes with 3 levels of difficulty and as players get used to one level of the game they can advance to the next level if they wish.

**Levels of spin:**

- This corresponds to no spin.
- This level will lead to the ball straightening at the 3rd or 6th column whichever being closer to the last person to touch the ball.
- This level will lead to the ball changing motion by from upwards to downwards or vice-versa at the 4th or 5th column whichever being closer to the last person to touch the ball.
- In this level the ball straightens at the 3rd or 6th column and then regains its original path of motion after crossing the 6th or 3rd column respectively.

**Levels of Difficulty:**

- This level corresponds to the normal game.
- In this level irrespective of the person who hit the ball the ball can rebound of both the 4th and the 5th column.
- In this level the ball has random possibility of rebounding from the 4th or 5th column whichever is closer to the last person to touch the ball.

**User Interface:**

**Inputs:**

1) 

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Paddle Of Player 2 remains Stationary</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Paddle Of Player 2 remains Stationary</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Paddle Of Player 2 moves Upward</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Paddle Of Player 2 moves Downward</td>
</tr>
</tbody>
</table>
2)

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Paddle Of Player 1 remains Stationary</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Paddle Of Player 1 remains Stationary</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Paddle Of Player 1 moves Upward</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Paddle Of Player 1 moves Downward</td>
</tr>
</tbody>
</table>

3) **INPUT1**:  
Change in the value of this switch is used to increment the amount of **Spin** being set by **Player 2** for the next time he hits the ball. This can take a maximum value of 3.

4) **INPUT2**:  
Change in the value of this switch is used to increment the amount of **Spin** being set by **Player 1** for the next time he hits the ball. This can take a maximum value of 3.

5) **CLEAR**:  
A zero value of this input shuts down the game. To restart the game this value should be reset to the value 1.

6) **LEV2, LEV3**:  
By setting LEV2 as high we will change the game to level 2. If both LEV2 and LEV3 are set then we are in level 3.
**Outputs:**

1) **8*8 LED matrix:**

This matrix is used as a display for the game. Each paddle is represented by two vertically aligned lit LED’s and the ball is represented by a single lit LED.

2) **LED array:**

The LED array is used to display the score of the game. LED’s 5 to 8 display the score for player 2 and LED’s 9 to 12 display the score for player 1. Scores are displayed in binary.
MODULAR CIRCUIT DIAGRAM

**Design description:**

1) **Module 1:**

This module is used to set whether the counter controlling the balls column number should be moving up or down. This module has a single D-flip flop containing the previous value that the up/down
control took. On the basis of this value it decides the next value of the up/down control.

In the normal level the output will remain the same as long as the ball has not collided with the paddle of a player. Upon colliding it changes its value of the up/down counter.

In level 2 it uses a PRBS to randomly decide to change the value of up/down when the ball is in the above defined columns. Similarly in the level 3 it uses the PRBS to change the value of the up/down counter at the above defined columns.

2) **Module 2:**

This is a module which chooses a speed for the ball motion depending on the value of the spin allocated to the ball. CLKMUX will take different values for different values of spin with the maximum value when spin is minimum and vice-versa. The possible frequencies for the ball to take are 15/8, 15/10, 15/12, 15/16 Hz.

3) **Module 3:**

This module is used to detect given the direction of motion will it collide with player 1 given by COL1 or with player 2 given by COL2.

Now COL1 will be 1 if BALLDIR is 1 and the ball row is either coincident with the upper row representing the paddle of player1 or is just above the paddle and is hence hitting the player1 at an angle from the top. It is also 1 if BALLDIR is 0 and the ball row is either coincident with the lower row representing the paddle of player1 or is just below the paddle and is hence hitting the player1 at an angle from the top or SIN is 1 and the ball is inline with the paddle.

COL2 shares the same relation with paddle representing player2.

4) **Module 4:**

This is a module to store the value of spin imparted to a ball when it gets rebounded of the paddle of a player. This will store the spin value set by whichever player has hit the ball.
5) **POSITION GENERATORS:**

This is a module which stores the value of the current position of the upper part of the paddle of a player in a 3 bit counter. For these the inputs A + not (B), C+ not(D) represents the up/down input while the input from module 5,6 forms the clock enable. The output POU represents whether the ball is in the row directly above the row or not. The output PODOWN represents whether the ball is in the row directly below the row or not. The output POOUT1 represents whether the ball is in the row directly in front of the upper row representing the paddle or not. The output POOUT2 represents whether the ball is in the row directly in front of the lower row representing the paddle or not.

6) **Module 5, 6:**

These modules are used to generate clock enable for the POSITION GENERATORS. This is generated every time the player must move up or down as long as there is a possibility for the player paddle to move in the direction required.

7) **PLONE SPIN, PLTWO SPIN:**

These modules are used to store the spin as being set by the player. Each time the input is toggled the spin counter is incremented by one with a maximum value of 3.

8) **Module 9:**

This Module is used to clear the spin registers storing the players set values for spin. The spin set by player 2 is cleared every time the ball either bounces of player 2 and crosses the 2\textsuperscript{nd} column and player 1 spin is cleared when the ball bounces of player 1 and crosses the 7\textsuperscript{th} column. Both spins will be cleared if a goal is scored at either end.

9) **Module 10:**

This is a module to store the score of the two players when either player scores a point his score is incremented. If either player scores 16 points then it will also put on the CHE signal to signal the end of the game.
EXPANDED MODULE 7

10) **Module 7:**

This module is used to give us the row of the ball at any given time. This consists of a counter to store the position of the row BALLROW, a DFF to store the direction in which the ball is moving, either up or down BALLDIR, and a DFF to store whether the ball is moving straight or not SIN.

The SIN signal becomes 1 if the ball is set to spin level 1 or 3 and the ball crosses the appropriate column. This signal is cleared whenever the ball reaches the column 7 or the column 1. Further at spin level 3 this pin is cleared whenever the ball crosses the appropriate column.

Whenever sin is set the BALLROW counter is switched off. Otherwise it is always on.

UP/DOWN for this counter depends upon on whether there is a collision with wall, in which case the value is changed or when there is a need to spin the ball, the value is also set to appropriate value after a collision with a player.
This value is set as per the following convention.

- If is hits the ball hits the edge of the paddle it is reflected backwards into the direction that it was moving.

- If is hits the ball hits the center of the paddle it is reflected along the same direction it arrived in.

- If the ball was moving straight if it hits the upper part of the paddle it is reflected upwards and if it hits the lower part of the paddle it is reflected downwards.

**Timing Simulation:**

1) Testing of the above circuit becomes extremely difficult by simulation as a graphic interface is required to input correct values for player up and down signals in order to get collisions with the ball and hence have useful test cases.

2) Hence most of the testing involving large portions of the circuit had to be done on the FPGA board so as to use the 8*8 matrix in order to view current states of the game and to enter proper inputs to test for the cases required.

3) However we have provided a timing and functional simulation of the entire system which confirm the working many of the various different cases that arise during the various modes of the game.
SOME TIMING SIMULATION RESULTS

IMPLEMENTATION RESULTS:

The device utilization summary of the circuit is:
- Total Number of CLB’s: 200 out of 400
- 4 input LUTs : 285 out of 800
- 3 input LUTs : 37 out of 400
- Number of External IOB’s: 49 out of 160

The timing analysis of the circuit is:
- Minimum period : 50.686ns
- Maximum frequency: 19.729MHz
- Maximum net delay: 21.749ns

**UCF Specification:**

```
NET A1 LOC=P14;
NET B1 LOC=P15;
NET C1 LOC=P16;
NET D1 LOC=P17;

NET INPF2 LOC=P27;
NET INPB2 LOC=P18;

NET CLRINP LOC=P23;

NET LEV2 LOC=P24;
NET LEV3 LOC=P25;

NET PLTWOSC<0> LOC = P81;
NET PLTWOSC<1> LOC = P82;
NET PLTWOSC<2> LOC = P83;
NET PLTWOSC<3> LOC = P84;
NET PLONESC<0> LOC = P3;
NET PLONESC<1> LOC = P4;
NET PLONESC<2> LOC = P5;
NET PLONESC<3> LOC = P6;

NET COLOUT<0> LOC= P60;
NET COLOUT<1> LOC= P61;
NET COLOUT<2> LOC= P62;
NET COLOUT<3> LOC= P65;
NET COLOUT<4> LOC= P56;
NET COLOUT<5> LOC= P57;
NET COLOUT<6> LOC= P58;
NET COLOUT<7> LOC= P59;

NET RA<0> LOC= P35;
NET RA<1> LOC= P45;
NET RA<2> LOC= P68;
NET RA<3> LOC= P38;
NET RA<4> LOC= P39;
NET RA<5> LOC= P67;
NET RA<6> LOC= P66;
NET RA<7> LOC= P69;

NET RB<0> LOC= P40;
NET RB<1> LOC= P47;
NET RB<2> LOC= P48;
NET RB<3> LOC= P49;
NET RB<4> LOC= P37;
NET RB<5> LOC= P36;
```
**Conclusion:**

**Limitations:**

- The game does not have a proper way to display the player scores.

- The current method of input lacks the ease of use which can be created if we were to use the pulse generators for input. However this would make it impossible to take any other input from the user such as spin, reset and level of the game.

- When speed of the ball changes the first motion of the ball occurs after an undefined value of time as the various clocks used as input to the speed multiplexer can not be synchronized.

- The game has no single player mode of functioning.

- The restricted size of the LED display puts a great constraint on the types of motion which can be executed by the ball to diagonal and straight motion.

**Refinements:**

- The output can be shown on seven segment displays using interface via the breadboard.

- We can incorporate a one player mode for the game.

- New levels can be added to the game for example a level in which scoring of a point makes a player invincible for the next four moves and other such increments.