COL215 – Mini Project
PING PONG GAME

Lab Partners – Samarth Aggarwal – 2016CS10395
- Ayush Patel – 2016CS10396

Specifications :

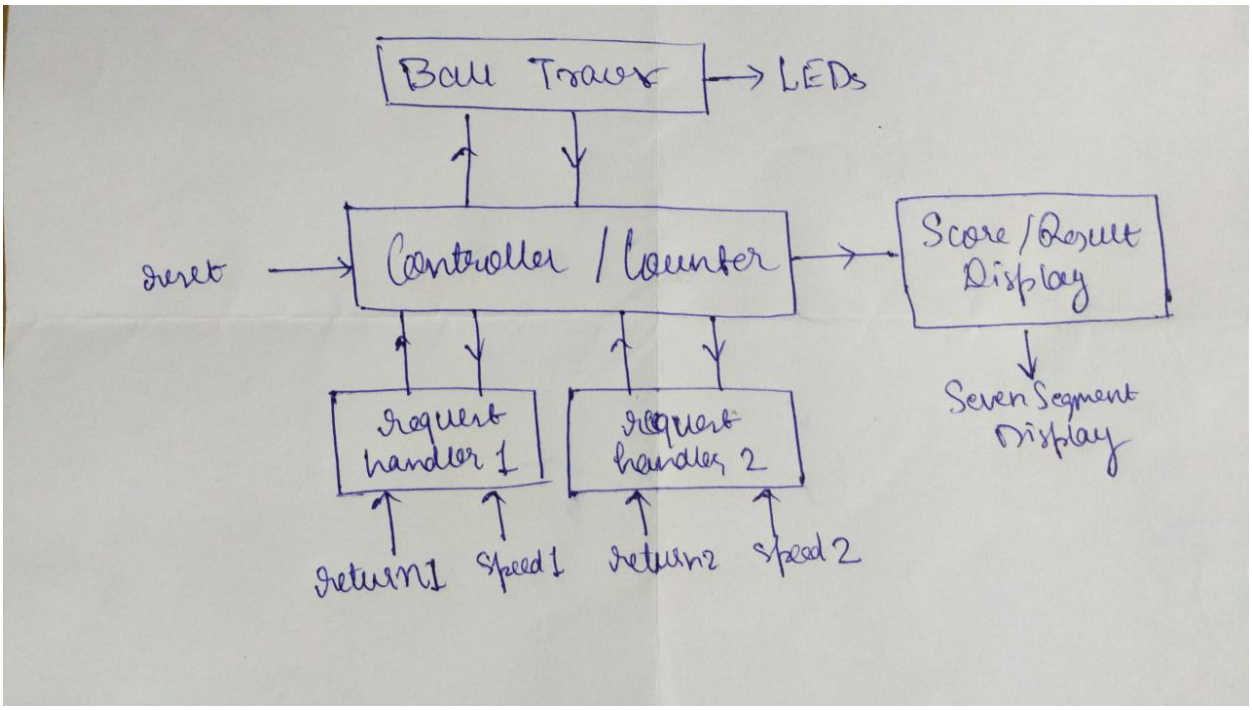
The project involves designing and implementation of “Ping – Pong Game” which is a two player game to be played on the FPGA basys board. The game involves a ball denoted by a lit LED that is returned by each player as it nears their side. The aim of this game is to defend one’s own goal post and score goals against the opponent by returning the ball as it approaches one’s goal post. The game ends as soon as one of thee player scores 9 goals against the other.

The 16 LEDs available on the basys board are used to display the position of the ball. Each player gets two push buttons to enter his requests. One of the pushbutton is used to return the ball when it has reached the end LED on his/her side. This button must be pressed only when the ball has has reached the last LED else a goal is scored. The other pushbutton is for telling the speed with which the ball is to be returned. The press of this button will toggle the speed with which the ball will be returned that is it dictates whether the ball is to be returned with a higher speed or with slower speed. Another button called reset has been introduced to set the initial values whenever a new game is to be started.

Overall Approach :

Since the press of a pushbutton is not recorded instantly and the circuit may make and break a few times before settling to the final value, hence we will use debouncing logic for the push buttons. For the debouncing logic, we will use another entity since many inputs have to be debounced so by creating another entity we can reuse the same code. Also, the reset button will work asynchronously so as to override the game at any stage and take it to the initial state of a new game. Also the time for which the ball remains at each of the 16 LED will be decided at the time of testing of the game since it has to be caliberated keeping in mind the ease of play, physical restrictions, etc. Similarily, the speeds for low and high speed modes will also be decided at the time of testing. Since two digits corresponding to the respecctive scores of the players as well as 2 letters corresponding tothe return speed of the players have to be displayed on the seven segment display, hence the whole display will be required. At the end of the game, 2 digits will depict the final score of the players while the rest of the 2 digits will denote which player has won the game.

Block Diagram Containing Major Subsystems:



1) Request Handlers – They take the requests from the pushbuttons and pass them on to the controller. The input to the request handler is not given directly from the buttons rather the output of the debouncing logic is fed into the request handler. This is done to prevent the recurring making and breaking of the circuit which is the root cause of why debouncing is needed.

2) Controller – This is the main component which processes the game. It registers the pushbutton requests and acts accordingly. Hence it checks whether the players have pressed the buttons at the right time or not. If yes, it turns the ball in the other direction and if no, it updates the score of the opponent player and continues the game. At the same time, it gives the output to the LEDs in order to display the position of the ball. Also, it maintains a track of the scoreboard and gives the output to the seven segment display accordingly. It is overriden asynchronously by the reset button which the initializes all the values.

3) Ball Tracer – It involves the LEDs that display the current position of the ball. It takes input from the controller and lits one of the LED at a time.

4) Score/Result Display – It refers to the seven segment display which displays the current score and also tell the player that won at the end of the game. Also, it denotes the speed with which the ball is to be returned by each player. At the end of the game, the final scores remain on the display while the player speeds are replaced by the notion of the winning player.

Test and Demonstration Plan :

The best way to test a game is by playing it. Hence, we plan to test it by playing it on the Xilinx basys board and rectifying the errors perceived.

Status of Coding :

The code for traversal of the ball ans reset functionalities had already been added in the preceding week. This week, the code for display on the seven segment display has been completed. Also, another entity has been added corresponding to the debouncing logic so as to facilitate reuse of the same code. Hence, debounce entity has been made. Also, the essential part of the controller and request handler have also been coded. Moreover, the debouncing logic has been integrated with the main entity. However, the display entity is yet to be integrated with the rest of the code.

Following is the description of the entity and architecture(containing declaration of signals) that we have used as a part of our VHDL code.

Entity:

```
entity ping_pong is
    Port ( return1 : in STD_LOGIC;
           return2 : in STD_LOGIC;
           speed1 : in STD_LOGIC;
           speed2 : in STD_LOGIC;
           reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           anode : out STD_LOGIC_VECTOR (3 downto 0);
           cathode : out STD_LOGIC_VECTOR (6 downto 0);
           led : out STD_LOGIC_VECTOR (15 downto 0));
end ping_pong;

entity debounce is
  Port (
    CLK : in  STD_LOGIC;
    x : in  STD_LOGIC;
    DBx : out  STD_LOGIC
  );
end debounce;

entity display is
          Port ( b : in STD_LOGIC_VECTOR (15 downto 0) :=(others => '0');
          clk : in STD_LOGIC :='0';
           cathode : out STD_LOGIC_VECTOR (6 downto 0) :=(others => '0');
```

```
        anode : out STD_LOGIC_VECTOR (3 downto 0) :=(others => '0'));
end display;
```

Architecture:

```
architecture Behavioral of ping_pong is
    signal count : INTEGER :=0;
    signal led_temp : STD_LOGIC_VECTOR (15 downto 0) :=(others => '0');
    signal dir : STD_LOGIC :='0';
    signal movel : STD_LOGIC :='0';
    signal mover : STD_LOGIC :='0';
    signal speed : STD_LOGIC :='0';


architecture Behavioral of debounce is
  type State_Type is (S0, S1);
  signal State : State_Type := S0;

  signal DPB, SPB : STD_LOGIC;
  signal DReg : STD_LOGIC_VECTOR (7 downto 0);



architecture Behavioral of display is
 signal count : INTEGER :=0;
 signal count2 : INTEGER :=0;
 signal cat0 : STD_LOGIC_VECTOR (6 downto 0) :=(others => '1');
 signal cat : STD_LOGIC_VECTOR (6 downto 0) :=(others => '1');
 signal cat1 : STD_LOGIC_VECTOR (6 downto 0) :=(others => '1');
 signal cat2 : STD_LOGIC_VECTOR (6 downto 0) :=(others => '1');
 signal cat3 : STD_LOGIC_VECTOR (6 downto 0) :=(others => '1');
 signal an : STD_LOGIC_VECTOR (3 downto 0) :=(others => '1');
```

Status of Testing :

The display has been tested independent of the main system. The glitches found were repaired and now the display seems to work perfectly. So far the testing phase for the game as a whole has not begun since we can test the game only once the overall system and display have been integrated together. However, the rest of the code has been made. All that is left to be done now is to integrate display and test our game. In the subsequent days, we plan to complete the testing phase of the game as a whole therby marking the successful completion of our project.