Implementation of Ping pong Game on FPGA

Jainam Jain
Department of Electronics and Communication Engineering
Nirma University
Ahmedabad
20bec043@nirmauni.ac.in

I. INTRODUCTION

The project is about one dimensional table tennis game. As the game on hardware implementation is something unique, it made us curious to go with it.
The game is very similar to real table tennis game. There are certain changes like rules, interface, etc. To be more specific the game is about precise timing.

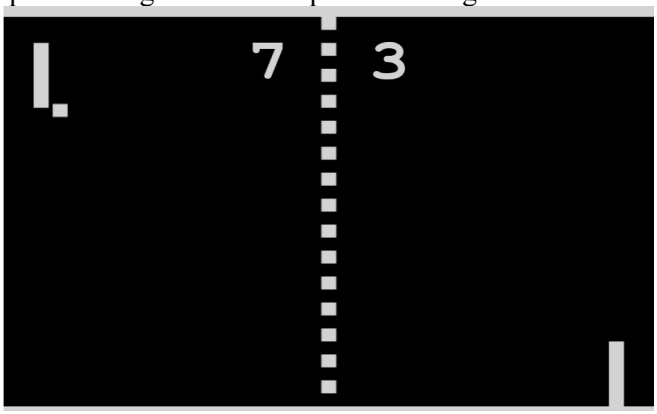


Figure – 1 - User interface of 2d ping pong game[online image] (reference: https://www.ponggame.org/)

This game closely resembles real table tennis, with some modifications to the rules and interface. The focus of the game is on precise timing, requiring players to have quick reflexes and a sharp eye. Through this project, we aimed to create a fun and engaging game that could be enjoyed by players of all skill levels
In this report, we will discuss the development process, challenges faced, and the final product. Implementing the glowing of sequential leds on an FPGA board makes it interesting to play for the user.

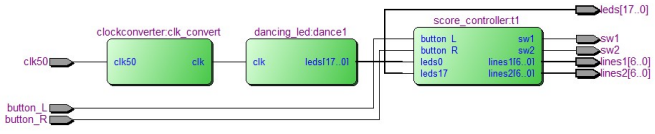
II. OBJECTTIVE

The objective of this project is to design and implement a one dimensional table tennis game on hardware, with a focus on precise timing. The game should closely resemble real table tennis, while also incorporating unique features and modifications to create an engaging experience for players. Our goal is to create a game that is challenging and enjoyable for players of all skill levels, while also providing a unique hardware implementation of a traditional game. In addition, we aim to document the development process and any challenges faced, to provide insights and lessons learned for future projects.

III. RULES AND BRIEFINGS

In one game, two players play at a time. The first player to reach 9 points wins the game. For each player, one playing button is assigned. Each of the 18 LEDs on the DE2 board will glow sequentially in the right and left directions.
When either of the extreme LED will glow, at that moment the player on that side has to press the switch with appropriate timing to score a point. If the player fails to press the switch with proper timing, he or she will not get the point. For implementing the project, simple Quartus software is used for dumping the code into the FPGA chip. Only the prior condition requires that existing code be erased from the FPGA chip.

IV. MAIN MODULE VIEW



THE SUBMODULES OF THE MAIN MODULE ARE AS FOLLOW :-

- "Score controller"
- "Clock converter"

- “Dancing leds”
- “BCD to seven segment display”

Port connection rules

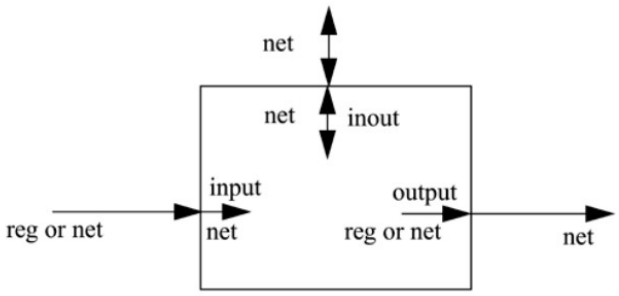
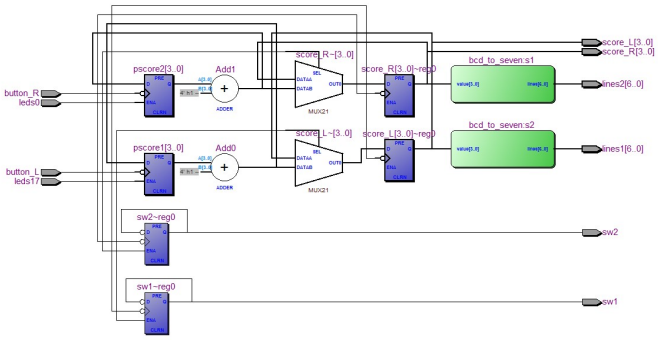


Diagram [Reference: Verilog HDL: A Guide to Digital Design and Synthesis- Book by samir palnitkar (pg no: 67)]

The input to any module can be of register or net data type but the input in the module is taken as net data type. Further the output of the module could be either of register data type or net data type.

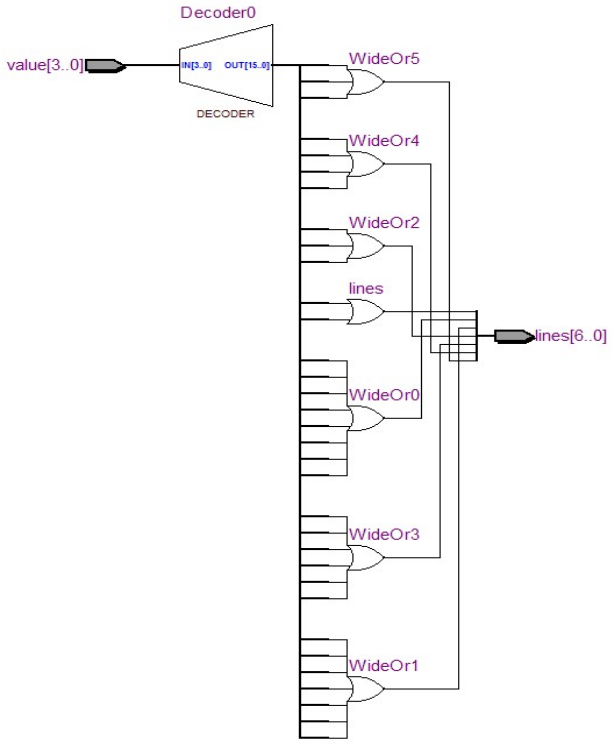
V. SCORE CONTROLLER MODULE

The module uses four inputs, namely button_L, button_R, leds0, and leds17, and two output registers: sw1, sw2, score_L, score_R, lines1, and lines2. Pscore1 and Psocre2 are the internal registers used for the dependencies of the code. This module is responsible for monitoring the score in the game. The module looks for the synchronisation of a pressed button and the respective led glowing at the extremes. For example, when the left led glows, the left button should be pressed simultaneously, and vice versa. Therefore, this module is responsible for checking the user input and the glowing of extreme leds. To carry out this task, we have used the negative edge of the pressed button. If the button is pressed synchronously, then at the end of the if else statement, scores are updated by converting the decimal value into a bcd value for the seven segment module.



VI. BCD TO SEVEN SEGMENT MODULE

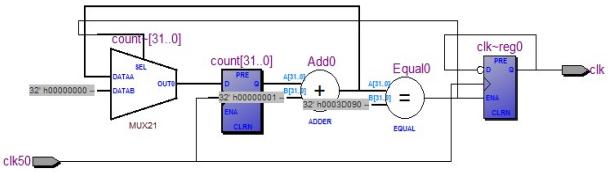
This module uses a 4-bit binary number as input and converts it into an ABCDEG, i.e., 7-segment display output. respective input binary number is shown on a 7-segment display. Both players can see their scores on the seven-segment displays allocated to them. This module gives those 7 leds their respective on and off states so that respective number can be displayed out there on the seven-segment display. Switch case statements are used to get the correct output on a 7-segment display.



RTL view of BCD to seven segment module

VII. CLOCK CONVERTER MODULE

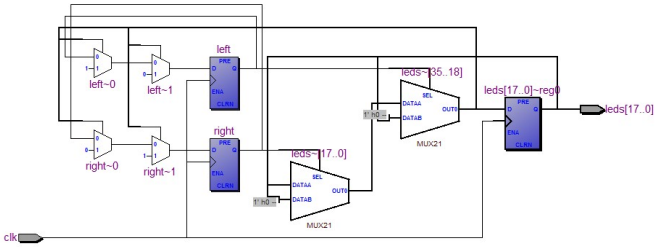
This module is used for converting the existing internal 50 MHz clock into our desired clock frequency. input to this module is a 50000000 hz clock, which is then converted to a lower frequency. (200 or 500 hertz) For implementing this module, a counter of 32 bits is used, which can go up to a maximum value of 232. This module is responsible for giving input to other modules. Many other modules use this clock as their reference. For example, dancing leds use this low-frequency clock for generating sequences of leds.



RTL view of clock converter module

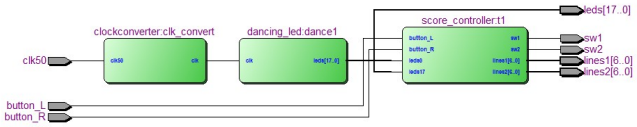
VIII. DANCING_LED MODULE

This module is used for glowing leds sequentially from right to left and left to right. The module operates on an input clock and outputs leds. From this module, the score controller uses the leds for led synchronisation. In the main module, the output of this module is used, more specifically, led 0 and led 17, to detect whether the button is pressed synchronously or not. For example, if the left button is pressed and the left extreme led is glowing simultaneously, then the score of the left player is to be increased by one.



RTL view of dancing_led module

IX. MAIN MODULE

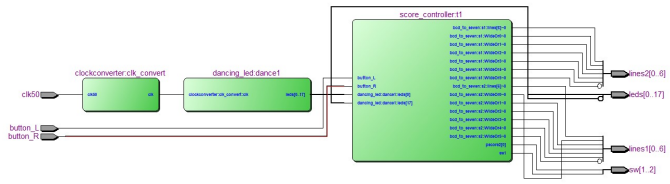


This module is responsible for interfacing all other submodules on one platform. In our case, submodules like dancing leds,clock converters, score controllers, and bcd to seven segments are interconnected for the execution of the final task. Additionally, wire clock and wire score_L and score_R are defined, which are used internally by the submodules. Score_L and Score_R are 4 bits wide. In this module, we have also kept sw1 and sw2 so that we can debug our code to see whether pressing buttons synchronously is giving output or not.

The flow of the main module is as stated in the register transfer level. The 50 Mhz clock is converted into a clock of lower frequency. This low-frequency clock is given to the dancing led module, which uses this clock to glow the leds sequentially from left to right and right to left. The output of the dancing_led module is given to the score controller, which also takes button_L and button_R as inputs and gives SW1 and SW2 and the score of each player along with sequential leds as output.

X. TECHNOLOGY MAP VIEWER OF THE PROJECT

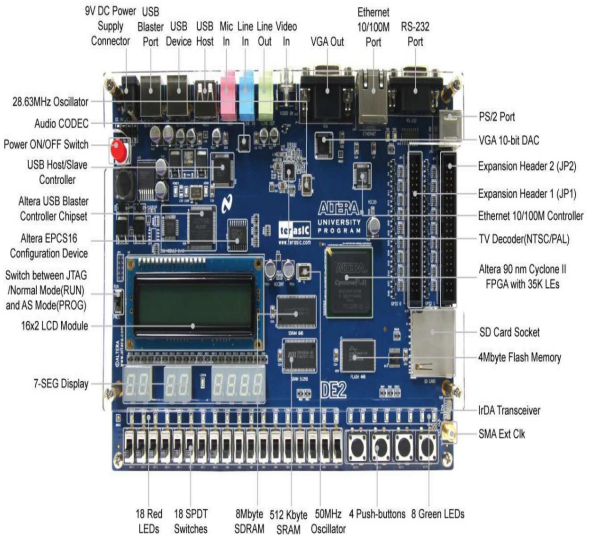
The Technology Map Viewer in Quartus Altera DE2 board is a powerful tool that enables the user to visualize the implemented circuit design in terms of its physical structure. It allows users to view and analyze the routing and placement of the different logic elements on the FPGA, providing a detailed look into the inner workings of the design.
The technology map viewer displays the physical representation of the design in the form of a two-dimensional matrix of logic elements. Each logic element is represented by a square, and the routing between the elements is indicated by lines connecting them. The user can zoom in and out of the design to get a closer look at specific areas of interest.



XI. RESULT OF SIMULATION WINDOW

FAMILY: CYCLONE II
DEVICE : EP2C35F672C6
TOTAL LOGIC ELEMENTS USED : 98
TOTAL COMBINATIONAL FUNCTIONS: 98
DEDICATED LOGIC REGISTERS : 61
TOTAL PINS : 37

XII. FPGA KIT ALONG WITH LABELS



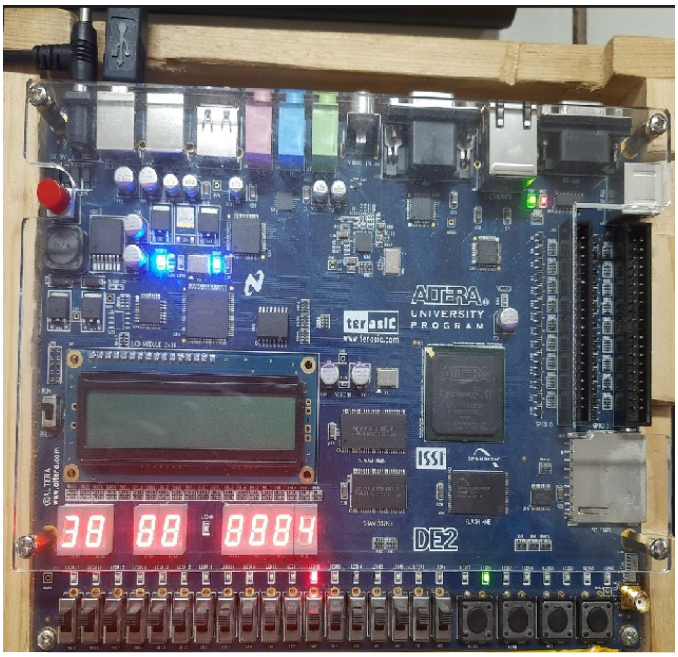
De2_Altera board [Image Reference: DE2_User_Manual (pg_no:4)]

XII. RESULTS

In this paper, we have successfully designed and implemented a Ping Pong game on an FPGA platform using Verilog hardware description language. The game is characterised by two buttons, 18 LEDs glowing sequentially from right to left and left to right, and a score counter. The design is optimised for maximum performance and uses hardware resources efficiently.

Our implementation provides a smooth and enjoyable gaming experience, with no lag or delays during gameplay. The game responds quickly to user input, and the score counter accurately updates the score.

In conclusion, our results demonstrate that the Ping Pong game can be successfully implemented on an FPGA platform using Verilog hardware description language, providing a fun and engaging gaming experience.
Our optimized design achieves high performance and resource utilization, making it a suitable platform for implementing similar games.



Abstract **- This paper presents the design and implementation of a Ping Pong game on a Field Programmable Gate Array (FPGA) platform. The game is designed using Verilog hardware description language and implemented on a De2 altera FPGA development board. The game is featured by two buttons with 18 leds glowing sequentially from right to left and left to right. The game also includes a score counter. The design is optimized for maximum performance and uses hardware resources efficiently. The implemented game provides a smooth and enjoyable gaming experience. The results show that the Ping Pong game can be successfully implemented on an FPGA platform and can provide a fun and engaging gaming experience.**

XIII. References

[1] S. Palnitkar, Verilog HDL, 2nd ed. Chennai: India Pearson Education Asia Pte. Ltd., 2013.

[2] IEEE Standard for Verilog Hardware Description Language. New York: IEEE, 2006

[3] Altera DE2 Development and Education Board - User Manual. California: Altera Corporation, 2006.

[4] Tutorial points simply easy learning www.tutorialspoint.com/vlsi_design/vlsi_design_verilogintroduction.htm2

APPENDIX

Code:

```
main_module(button_L,button_R,clk50
,leds,lines1,lines2,sw1,sw2);
input button_L,button_R,clk50;
output [17:0]leds;
output [6:0]lines1,lines2;
output sw1,sw2;

wire clk;
wire [3:0]score_L,score_R;

//interconnecting all modules
clockconverter clk_convert(clk50,clk);
dancing_led dance1(clk,leds);
score_controller
t1(button_L,button_R,leds[0],leds[17],
sw1,sw2,score_L,score_R,lines1,lines2
);
endmodule

module bcd_to_seven(value,lines);
input [3:0] value;
output reg[6:0]lines;
always @(*)
begin
 case (value)
    4'd0: lines=7'b0000001;
4'd1: lines=7'b1001111;
    4'd2: lines=7'b0010010;
    4'd3: lines=7'b0000110;
    4'd4: lines=7'b1001100;
    4'd5: lines=7'b0100100;
```

```
4'd6: lines=7'b0100000;
    4'd7: lines=7'b0001111;
    4'd8: lines=7'b0000000;
    4'd9: lines=7'b0000100;
    default: lines= 7'b0110000;
 endcase
end
endmodule

module
score_controller(button_L,button_R,le
ds0,leds17,
sw1,sw2,score_L,score_R,lines1,lines2
);
input button_L,button_R;
input leds0,leds17;
output reg sw1,sw2;
output reg [3:0]score_L=4'b0000;
output reg [3:0]score_R=4'b0000;
output [6:0] lines1,lines2;
reg [3:0]pscore1=4'b0000;
reg [3:0]pscore2=4'b0000;

always @(negedge button_L)
begin

if( (leds17) )
    begin
        score_L=pscore1+4'b0001;
    pscore1=score_L;
        sw1=~sw1;
     end
end
always @(negedge button_R)
begin
    if((leds0))
        begin
            score_R=pscore2+4'b0001;
            pscore2=score_R;
            sw2=~sw2;
        end
end
bcd_to_seven s2(score_L,lines1);
bcd_to_seven s1(score_R,lines2);
endmodule

module clockconverter(input
clk50,output reg clk);
reg[31:0]count;

initial
begin
 count=32'd0;
 clk=0;
end

always @(posedge clk50)
begin
    count=count+1;
    if(count==32'd5000000)
    begin
        clk=~clk;
        count=32'd0;
    end
end
endmodule

module dancing_led(clk,leds);
input clk;
output reg [17:0] leds;
```

```
reg left,right;

initial
begin
    leds=18'b1;
    left=1'b1;
    right=1'b0;
end

always @(posedge clk)
begin
    if(left==1)
        leds=leds<<1;
    else if(right==1)
        leds=leds>>1;

    if( leds[0] )
    begin
        left=1'b1;
        right=1'b0;
    end
    if(leds[17])
    begin
        left=1'b0;
        right=1'b1;
    end
end
endmodule
```