Merve Barın

Doğanay Şirintuna

ELEC LAB PROJECT REPORT

**Introduction**

In this project we were asked to implement a two-paddle game in which the Ping-Pong ball is modeled by sequential illumination of the LEDs in two directions. The paddles are push buttons, which must be pressed at just the right time to return the ball.

**A Brief Explanation of the Game**

In order to show the moving ball, we used 8 different Leds. We used two push buttons for the paddles of the users. In order for the game to start the users first push their button at the same time. Both of the users have the right to serve the ball twice in a row then the serve right changes. If a user pushes the button at the right time, the ball changes the direction, if a user pushes at the wrong time, the other user gets the point and the score board updates itself. The game finishes when one of the user’s score reaches 10. When a user’s score reaches 10, all leds turn on then off. We added a reset button in order to stop the game and start it from the beginning. When user 1 wins the seven-segment display part of the FPGA board shows 1111, which means user 1 has won. If user 2 wins then the board shows 2222, which means user 2 has won the game.

**TRUTH TABLE FOR COUNTER**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D | C | B | A |  |  | D+ | C+ | B+ | A+ |
| 0 | 0 | 0 | 0 |  |  | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 |  |  | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 |  |  | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |  |  | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |  |  | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 |  |  | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 |  |  | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |  |  | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |  |  | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 |  |  | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 |  |  | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 |  |  | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |  |  | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 |  |  | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 |  |  | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |  |  | 0 | 0 | 0 | 0 |

We used four bits to show FF. One bit is for the direction, other three is for the eight different led configurations. We added a clock cycle for our counter to work properly in order to make the state changes in leds.

**KARNOUGH MAP FOR D**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DC/BA | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  | 1 |  |
| 11 | 1 | 1 |  | 1 |
| 10 | 1 | 1 | 1 | 1 |

D+ = DB’ + DC’ + DA’ + D’CBA

D+ = D(BCA)’ + D’(CBA)

D+ = D xor (B and C and A)

In our code FF(3) equals D.

**KARNOUGH MAP FOR C**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DC/BA | 00 | 01 | 11 | 10 |
| 00 |  |  | 1 |  |
| 01 | 1 | 1 |  | 1 |
| 11 | 1 | 1 |  | 1 |
| 10 |  |  | 1 |  |

C+ = CB’ + CA’ + C’BA

C+ = C(BA)’ + C’(BA)

C+ = C xor (B and A)

In our code FF(2) equals C.

**KARNOUGH MAP FOR B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DC/BA | 00 | 01 | 11 | 10 |
| 00 |  | 1 |  | 1 |
| 01 |  | 1 |  | 1 |
| 11 |  | 1 |  | 1 |
| 10 |  | 1 |  | 1 |

B+ = B’A + BA’

B+ = B xor A

In our code FF(1) equals B.

**KARNOUGH MAP FOR A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DC/BA | 00 | 01 | 11 | 10 |
| 00 | 1 |  |  | 1 |
| 01 | 1 |  |  | 1 |
| 11 | 1 |  |  | 1 |
| 10 | 1 |  |  | 1 |

A+ = B’A’ + BA’

A+ = A’

In our code FF(0) equals A.

**LED REPRESENTATION TABLE**

|  |  |  |
| --- | --- | --- |
|  | Representative States | Led Configurations |
| Edge | 0001 | 10000000 |
|  | 0010 | 01000000 |
|  | 0011 | 00100000 |
| Middle | 0100 | 00010000 |
|  | 0101 | 00001000 |
|  | 0110 | 00000100 |
|  | 0111 | 00000010 |
| Edge | 1000 | 00000001 |
|  | 1001 | 00000010 |
|  | 1010 | 00000100 |
| Middle | 1011 | 00001000 |
|  | 1100 | 00010000 |
|  | 1101 | 00100000 |
|  | 1110 | 01000000 |
| Error | 1111 | 11111111 |
| End | others (0000) | 00000000 |

**Conclusion**

In this project we learned how to display four different number independently on a FPGA board and updating it when one of the user scores. We used four bit variable to show different led configurations. To make the changes in this states in the order which we wanted and to be able to see all four segment of the seven-segment display part of the FPGA board, we used clock cycle. This game starts when two paddles, which are the middle push buttons on the FPGA board, pushed at the same time. The ball, the first led, starts from left and goes to right with time. If right user pushes at the right time the ball starts going left. If right one can’t push the button at the right time, left user gets the point and scoreboard updates the scores. The ball starts from left twice, then starts from right twice and this goes on until one of the users reaches 10 points. When someone wins the scoreboard shows which of the user has won. While writing the code we had some issues while showing the scores on the scoreboard because we first couldn’t find a way to change each segment of the board independently. We have overcome this issue with using a counter and a clock. The lights of the each segment turn on and off so quickly that our eyes couldn’t see when the lights were turned off. So all four display parts looks as turned on at the same time but changes independently.

**Our VHDL Code for the Game**

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating

---- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ledpingpongcode is

port(

signal CLK: in STD\_LOGIC;

signal clk1: in STD\_LOGIC;

signal Fleds: out STD\_LOGIC\_VECTOR(7 downto 0);

signal Reset: in STD\_LOGIC:='0';

signal User1: in STD\_LOGIC:='0';

signal User2: in STD\_LOGIC:='0';

signal segment\_disp :out std\_logic\_vector(6 downto 0);

signal display: out std\_logic\_vector(3 downto 0));

end ledpingpongcode;

architecture Behavioral of ledpingpongcode is

signal score:STD\_LOGIC\_VECTOR(1 downto 0);

signal R04:STD\_LOGIC\_VECTOR(3 downto 0);

signal R14:STD\_LOGIC\_VECTOR(3 downto 0);

signal FF: STD\_LOGIC\_VECTOR(3 downto 0):="0000";

signal R0, R1,L0,L1,sayac,serve: integer :=0;

signal state\_display:STD\_LOGIC\_VECTOR(3 downto 0);

signal state :STD\_LOGIC\_VECTOR(3 downto 0);

signal L04:STD\_LOGIC\_VECTOR(3 downto 0);

signal L14:STD\_LOGIC\_VECTOR(3 downto 0);

begin

process(CLK)

begin

if(rising\_edge(CLK)) then

if(FF="0000" and User1='1' and User2='1' and serve<2)then

serve<=serve+1;

FF<="0010";

end if;

if (Reset='1') then

R1<=0;

R0<=0;

L1<=0;

L0<=0;

serve<=0;

end if;

if (FF="1111")then

FF<="0000";

end if;

if(FF="0000" and User1='1' and User2='1' and 1<serve)then

serve<=serve+1;

FF<="1001";

if(2<serve) then

serve<=0;

end if;

end if;

if(((FF = "0000" or FF = "0010" or FF = "0011" or FF = "0100" or FF = "0101" or FF = "0110" or FF = "0111" or FF = "1001" or FF = "1010" or FF = "1011" or FF = "1100" or FF = "1101" or FF = "1110") and User1='0' and User2='0' ) or (FF="1000" and (User1='0' and User2='1')) or (FF="0001" and (User1='1' and User2='0'))) then

FF(3)<=(FF(2) and FF(1) and FF(0)) xor FF(3);

FF(2)<=(FF(1) and FF(0)) xor FF(2);

FF(1)<=(FF(0)) xor FF(1);

FF(0)<=not(FF(0));

end if;

if (FF="0001" and User2='1') then

FF<="1111";

L0<=L0+1;

if(L0>8) then

R1<=1;

R0<=1;

L1<=1;

L0<=1;

end if;

end if;

if (FF="0001" and User1='0') then

FF<="1111";

R0<=R0+1;

if(R0>8) then

R1<=2;

R0<=2;

L1<=2;

L0<=2;

end if;

end if;

if(FF="1000" and User1='1')then

FF<="1111";

R0<=R0+1;

if(R0>8) then

R1<=2;

R0<=2;

L1<=2;

L0<=2;

end if;

end if;

if (FF="1000" and User2='0') then

FF<="1111";

L0<=L0+1;

if(L0>8) then

R1<=1;

R0<=1;

L1<=1;

L0<=1;

end if;

end if;

if((FF = "0010" or FF = "0011" or FF = "0100" or FF = "0101" or FF = "0110" or FF = "0111" or FF = "1001" or FF = "1010" or FF = "1011" or FF = "1100" or FF = "1101" or FF = "1110" or FF = "1111") and (User1='1' and User2='0'))then

FF<="1111";

R0<=R0+1;

if(R0>8) then

R1<=2;

R0<=2;

L1<=2;

L0<=2;

end if;

end if;

if((FF = "0010" or FF = "0011" or FF = "0100" or FF = "0101" or FF = "0110" or FF = "0111" or FF = "1001" or FF = "1010" or FF = "1011" or FF = "1100" or FF = "1101" or FF = "1110" or FF = "1111") and (User1='0' and User2='1'))then

FF<="1111";

L0<=L0+1;

if(L0>8) then

R1<=1;

R0<=1;

L1<=1;

L0<=1;

end if;

end if;

if(FF="1110" and User1='0' and User2='0') then

FF<="0001";

end if;

if (FF="0000" and (User1='0' or User2='0')) then

FF<="0000";

end if;

end if;

R04<=conv\_std\_logic\_vector(R0,4);

R14<=conv\_std\_logic\_vector(R1,4);

L04<=conv\_std\_logic\_vector(L0,4);

L14<=conv\_std\_logic\_vector(L1,4);

end process;

------------------------------

process(clk1)

begin

if(rising\_edge(clk1)) then

sayac <=sayac+1;

if(sayac = 100) then

score(1 downto 0) <= score(1 downto 0) + 1 ;

sayac<=0;

case score(1 downto 0) is

when "00" => state <=R04 ; state\_display <= "1110" ;

when "01" => state <=R14; state\_display <= "1101" ;

when "10" => state <=L04 ; state\_display <= "1011" ;

when others => state <=L14 ; state\_display <= "0111" ;

end case;

case state is

when "0000"=>segment\_disp<= "1000000";

when "0001"=>segment\_disp<= "1111001";

when "0010"=>segment\_disp<= "0100100";

when "0011"=>segment\_disp<= "0110000";

when "0100"=>segment\_disp<= "0011001";

when "0101"=>segment\_disp<= "0010010";

when "0110"=>segment\_disp<= "0000010";

when "0111"=>segment\_disp<= "1111000";

when "1000"=>segment\_disp<= "0000000";

when "1001"=>segment\_disp<= "0010000";

when others=>segment\_disp<= "0000000";

end case ;

display<= state\_display;

end if;

end if;

end process;

-------------------------------------------

With FF select

Fleds<="10000000" when "0001",

"01000000" when "0010",

"00100000" when "0011",

"00010000" when "0100",

"00001000" when "0101",

"00000100" when "0110",

"00000010" when "0111",

"00000001" when "1000",

"00000010" when "1001",

"00000100" when "1010",

"00001000" when "1011",

"00010000" when "1100",

"00100000" when "1101",

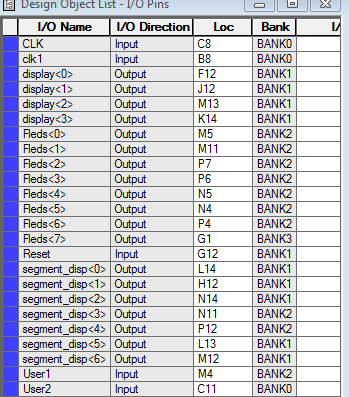
"01000000" when "1110",

"11111111" when "1111",

"00000000" when others;

end Behavioral;

**Our UCF File for the Game**

****