Lab 1: FPGA-based Mental Binary Math Game

Jose Navarro

ECE5440

**Introduction:**

The Mental Binary Game is a simple two player game testing the players ability to convert hexadecimal numbers into binary numbers. In this game player one will think of a binary number and input the number to the FPGA using the FPGA switches. Without looking at player one’s switches, player two is to look at the FPGA’s seven segment decoder displaying player one’s number in hexadecimal. Player two is then to input the number, using the switches on the FPGA board, they believe will sum with player one’s number to equal 15. Player two’s goal is to mentally come up with the number that when added to player one’s number the total sum will equal 15 (or F when displayed on the FPGA). Player one’s goal is to come up with a number difficult enough so that player two is not able to come up with the right number to match the original output. If player two comes up with the correct number to match player one’s number player two gets a point. If player two does not come up with the correct number player one gets a point. Players change roles after each point is scored and play as long as they desire.

**System Architecture Design:**

Figure 1 below shows the top-level system architecture.

A screenshot of a cell phone

Description automatically generated

*Figure 1. Top-Level System Architecture*

This module’s function is to take two 4-bit binary inputs, sum them together and decode them to properly display both sum and inputs to a seven segment display as shown in the figure above.

**Simulation Results:**

Figure 2 below shows the simulation results of the adder. As shown the adder properly adds 3 different cases of 4-bit inputs. The module was written when given inputs in binary and decimal values. As shown by the simulation the adder module works as expected.

A screenshot of a computer

Description automatically generated

*Figure 2. Test Bench Adder simulation*

Figures 3 and 4 below shows the simulation results of the seven-segment decoder. The figure shows all possible decoded values (decoder is only designed to output 4-bit numbers in hexadecimal to a single seven segment decoder). As shown by the simulation figures the seven segment decoder works as expected.

A screenshot of a computer

Description automatically generated

*Figure 3. Test Bench Seven-Segment Decoder simulation Pt. 1*

**A screenshot of a computer

Description automatically generated**

*Figure 4. Test Bench Seven-Segment Decoder simulation Pt. 2*

**FPGA Board Testing Results:** Figures 5 and 6 show the FPGA when player two matches the correct number for player one’s input and when player two does not match the correct number for player one’s input. As shown by the figures when player two correctly matches the number to produce a sum of 15 (F on the seven-segment decoder) a green LED is lit up. Otherwise a red LED is lit up, indicating that player 2 is incorrect.

A circuit board

Description automatically generated

*Figure 5. FPGA Board, Player 2 input number does not result in a sum of 15 given Player 1 number*

A circuit board

Description automatically generated

*Figure 6. FPGA Board, Player 2 input number results in a sum of 15 given Player 1 number*

**Video Demo:**

<https://drive.google.com/drive/u/0/folders/1yVXeZbkj3hUiTtPaAfvST8fmr7XfrWuP>

**Conclusion:**

The Mental Binary Math Game is a simple two player game that sharpens one’s ability to add binary numbers mentally. It is a very simple game to implement using modules and the FPGA board. The lab is fully completed including the bonus feature. The bonus feature indicates when player two has matched player one’s binary number and when they have not using a green LED when the number is matched and a red LED when the number is not matched.

**Appendix:**adder\_top.v :

module adder\_top(switch1, switch2, seg1, seg2, segSum, red, green);

input [3:0] switch1, switch2;

output [6:0] seg1, seg2, segSum;

output red, green;

wire [3:0] sum;

adder adder\_1 (switch1, switch2, sum);

sevenSegDecoder segNum1(switch1, seg1);

sevenSegDecoder segNum2(switch2, seg2);

sevenSegDecoder segSumation(sum, segSum);

ledToggle led(sum, red, green);

endmodule

adder.v :

module adder(num1, num2, sum);

input [3:0] num1, num2;

output [3:0] sum;

reg [3:0] sum;

always @ (num1, num2)

begin

sum = num1 + num2;

end

endmodule

adder\_tb.v :

`timescale 10ns/100ps

module adder\_tb();

reg [3:0] in1, in2;

wire [3:0] out1;

adder DUT\_adder(in1, in2, out1);

initial

begin

in1 = 0;

in2 = 0;

#10;

in1 = 1;

in2 = 2;

#10;

in1 = 4'b0011;

in2 = 4'b1111;

end

endmodule

sevenSegDecoder.v :

module sevenSegDecoder(numIn, numOut);

input [3:0] numIn;

output [6:0] numOut;

reg [6:0] numOut;

always @ (numIn)

begin

case(numIn)

4'b0000: begin numOut = 7'b1000000; end

4'b0001: begin numOut = 7'b1111001; end

4'b0010: begin numOut = 7'b0100100; end

4'b0011: begin numOut = 7'b0110000; end

4'b0100: begin numOut = 7'b0011001; end

4'b0101: begin numOut = 7'b0010010; end

4'b0110: begin numOut = 7'b0000010; end

4'b0111: begin numOut = 7'b1111000; end

4'b1000: begin numOut = 7'b0000000; end

4'b1001: begin numOut = 7'b0011000; end

4'b1010: begin numOut = 7'b0001000; end

4'b1011: begin numOut = 7'b0000011; end

4'b1100: begin numOut = 7'b1000110; end

4'b1101: begin numOut = 7'b0100001; end

4'b1110: begin numOut = 7'b0000110; end

4'b1111: begin numOut = 7'b0001110; end

default: begin numOut = 7'b1111111; end

endcase

end

endmodule

sevenSegDecoder\_tb.v :

`timescale 10ns/100ps

module sevenSegDecoder\_tb();

reg [3:0] inputPin;

wire [6:0] outputSeg;

sevenSegDecoder DUT\_sevenSegDecoder(inputPin, outputSeg);

initial

begin

inputPin = 4'b0000;

#10;

inputPin = 4'b0001;

#10;

inputPin = 4'b0010;

#10;

inputPin = 4'b0011;

#10;

inputPin = 4'b0100;

#10;

inputPin = 4'b0101;

#10;

inputPin = 4'b0110;

#10;

inputPin = 4'b0111;

#10;

inputPin = 4'b1000;

#10;

inputPin = 4'b1001;

#10;

inputPin = 4'b1010;

#10;

inputPin = 4'b1011;

#10;

inputPin = 4'b1100;

#10;

inputPin = 4'b1101;

#10;

inputPin = 4'b1110;

#10;

inputPin = 4'b1111;

#10;

end

endmodule

ledToggle.v :

module ledToggle(sum, red, green);

input [3:0] sum;

output red, green;

reg red, green;

always @ (sum)

begin

if (sum == 15)

begin

red = 0;

green = 1;

end

else

begin

red = 1;

green = 0;

end

end

endmodule