**Lab 1: FGPA-based Mental Binary Math Game**

**Stefan Bucur**

**ECE5440**

**1. Introduction:**

The Binary Math Game is a two-player turn-based game that can test the player’s ability to interpret binary and hexadecimal numbers and add them. The players decide who goes first in the first round. That person enters a binary number into one of the sets of switches which is displayed on the closest hex display. The other tries to enter another binary number into the second set of switches which is displayed on another hex display next to them (the second player is expected not to peek at the other player’s switches during this time). A third hex display shows the two numbers added together. There are a pair of LEDs next to the sum display indicating if the number correctly adds to 15: green for a win, and red for a loss. Each round players can change turns and keep score by their own means.

**2. System Architecture Design:**

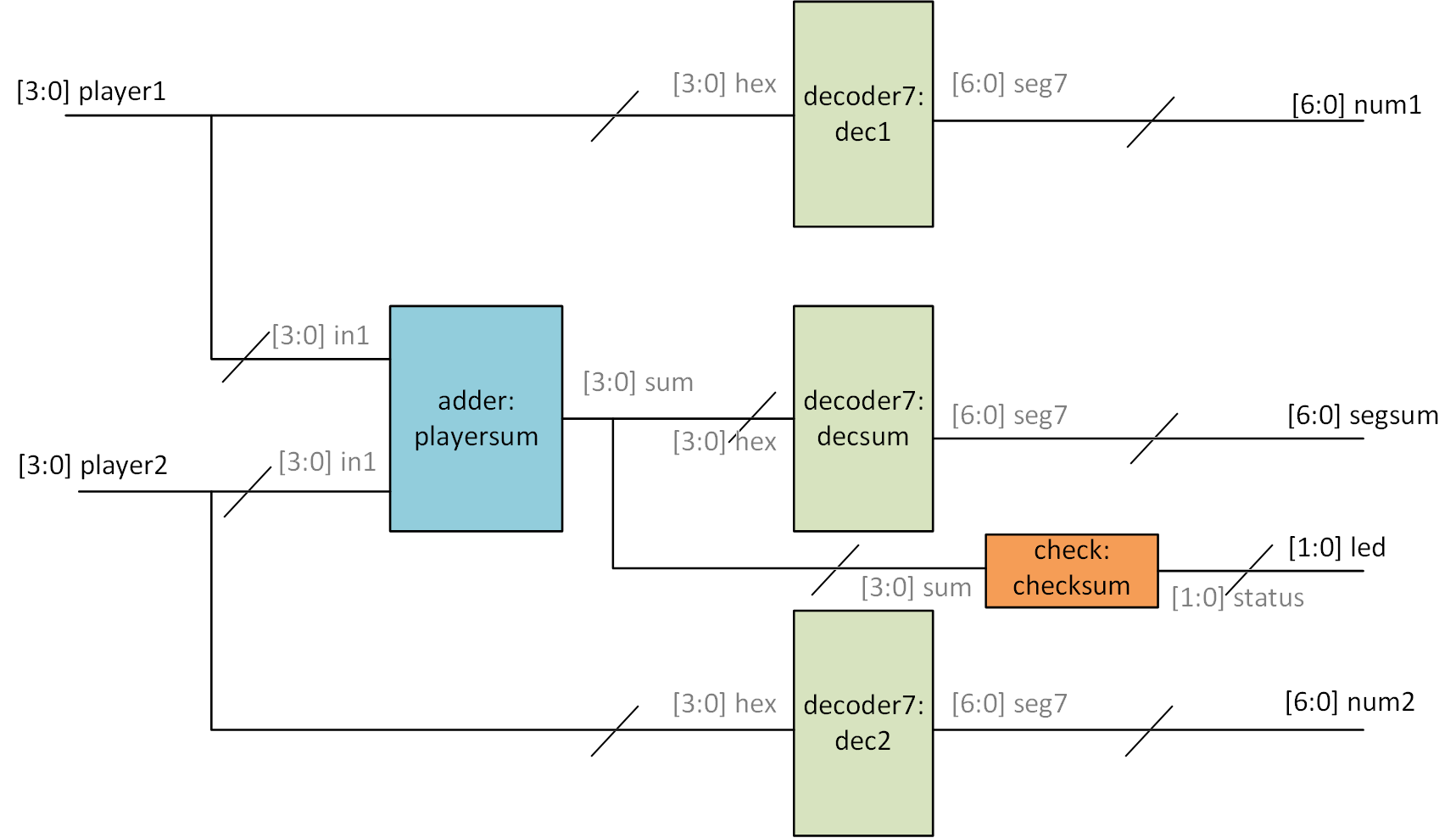


Figure 1: top-level module design of the game. Gray input/output names denote their respective submodule variables.

The top-level module uses three decoders, each separately wired to a seven-segment display meant to display the input value as a hexadecimal digit. The decoder used to the display the sum has its input connected to the adder’s output. Each of the two player’s set of 4 switches is connected to their corresponding decoder and the two inputs for the adder. The adder module performs 4-bit addition with no additional carry bit. The check module needs to output to two separate LEDs, with status[0] intended for a green LED and status[1] for red. The entire system is built on combinational logic, so no clock signals are needed, and all operations are done as soon as the signals arrive to the modules. Players will see the display update every time a switch changes state.

**3. Simulation Results:**

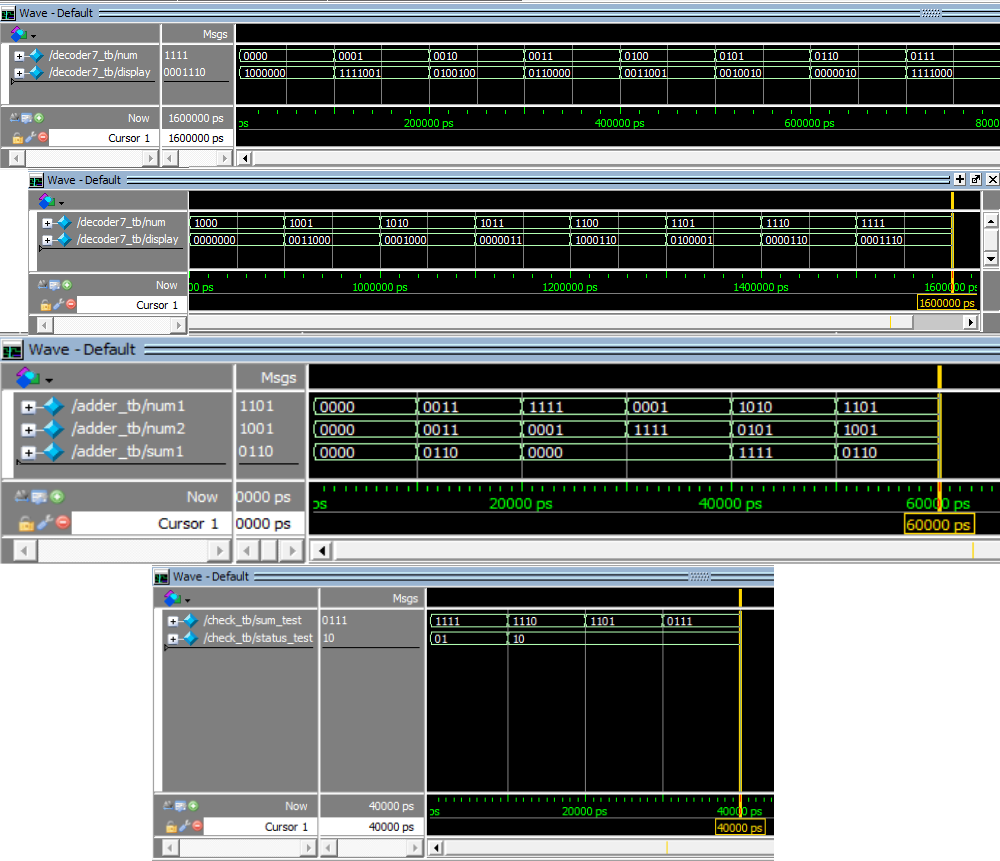


Figure 2: simulation of the decoder7 (top), adder (middle), and check module (bottom). The decoder7 and check modules were tested on a 10ns/100ps timescale, while the adder testbench used a 1ns/100ps timescale.

The decoder7 module was tested for all 16 different inputs since the set of all possible values was small enough. Adder has too many for the same treatment (256 possible inputs), so test cases were carefully chosen to test for sums below, at, or above the winning value of 15. The check module was tested for all 4 inputs, where the output signal values swap to turn on a green LED only when the input is 1111.

**4. FPGA Board Testing Results:**

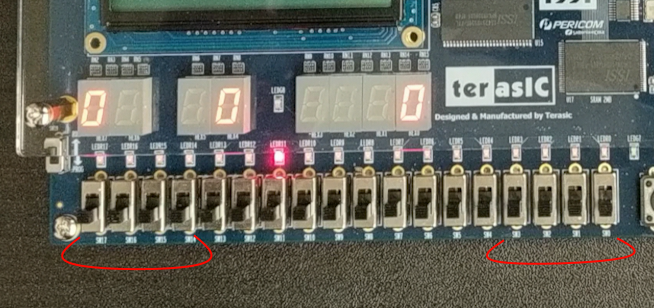


Figure 3: Initial state of the game. The red lines indicate the two sets of four switches configured to be used by the two players.

When the FPGA is first powered up, the programmed logic immediately begins to display the inputs and the sum onto the hex displays. The player1 and num1 wires are pinned out to the left four switches and hex display; player2 and num2 have the rightmost switches and display. Note the sum in Figure 3 is not 15, so the red LED connected to the check module is on.

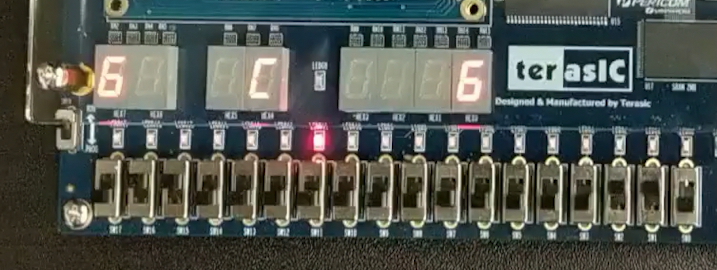


Figure 4: Losing outcome for a single round.

The round in Figure 4 ends in a loss since 6+6 = 12, which is less than 15. Note the red LED has not changed state. Players playing multiple rounds would keep score using their own means; the game logic currently does not support scorekeeping.

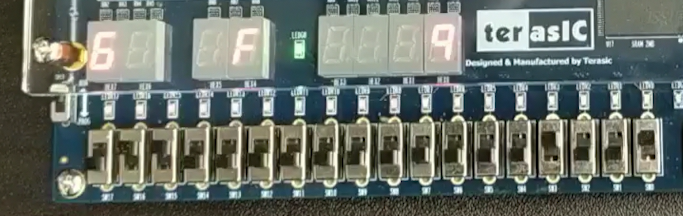


Figure 5: Winning outcome for a single round.

Here player 2 was able to figure out the sum and get the sum display to show 15 in hexadecimal. This indicates a win to the check module, so now only the green LED is on. Again, players can record the score themselves or start over.

**5. Video Demo:**

A short video demonstration of the game working on the FPGA can be watched by clicking on [this Google Drive link](https://drive.google.com/file/d/1zrRQ_FSfEsCcr2s6faFDjCpv9vGexMYc/view?usp=sharing). The demonstration shows how two players can enter numbers into the board, showing both a winning and losing match.

**6. Conclusion:**

The Verilog code was successfully compiled and flashed to the FPGA, where the board behaved as expected from the simulations in Figure 2. The bonus check module was also completed to visually indicate when a player would win or lose.

**7. Appendix:**

**// Lab1\_BUCUR\_S.v:**

module Lab1\_BUCUR\_S(

//inputs

player1, player2,

//outputs

num1, num2, segsum, led);

input [3:0] player1, player2;

output [6:0] num1, num2, segsum;

output [1:0] led;

wire [3:0] sum;

wire [1:0] led;

adder playersum(player1, player2, sum);

decoder7 dec1(player1, num1);

decoder7 dec2(player2, num2);

decoder7 decsum(sum, segsum);

check checkmatch(sum, led);

endmodule

**// decoder7.v:**

module decoder7 (hex, seg7);

input [3:0] hex;

output [6:0] seg7;

reg [6:0] seg7;

always @ (hex, seg7)

begin

case(hex)

4'b0000: begin seg7 = 7'b1000000; end

4'b0001: begin seg7 = 7'b1111001; end

4'b0010: begin seg7 = 7'b0100100; end

4'b0011: begin seg7 = 7'b0110000; end

4'b0100: begin seg7 = 7'b0011001; end

4'b0101: begin seg7 = 7'b0010010; end

4'b0110: begin seg7 = 7'b0000010; end

4'b0111: begin seg7 = 7'b1111000; end

4'b1000: begin seg7 = 7'b0000000; end

4'b1001: begin seg7 = 7'b0011000; end

4'b1010: begin seg7 = 7'b0001000; end

4'b1011: begin seg7 = 7'b0000011; end

4'b1100: begin seg7 = 7'b1000110; end

4'b1101: begin seg7 = 7'b0100001; end

4'b1110: begin seg7 = 7'b0000110; end

4'b1111: begin seg7 = 7'b0001110; end

endcase

end

endmodule

**// adder.v:**

module adder (in1, in2, sum);

input [3:0]in1, in2;

output [3:0] sum;

reg [3:0] sum;

always @ (in1, in2)

begin

sum = in1 + in2;

end

endmodule

**// check.v:**

module check(

//inputs

sum,

//outputs

status);

input [3:0] sum;

output [1:0] status;

reg [1:0] status;

always @(sum)

begin

if( sum == 4'b1111 )

begin

status = 2'b01;

end

else

begin

status = 2'b10;

end

end

endmodule