CompE470L Project Report Draft

***Memorization Game using FPGA***

***Demoed in lab period on 12/6/17***

**Project Description:**

My project is a memorization game using non-fixed memory size and colored LED’s matched with pushbuttons. This will be implemented using the standard FPGA board used for all labs in this course. The project has the following features:

* 4-bit LED sequence displaying the correct bit to be memorized
* LED’s mapped to pushbuttons via color-coding
* Score multiplexing

The program creates increasingly difficult memorization sequences based on how long the player has been playing. For example, when the player first starts, one random LED will be lit for a short time. If the player presses the corresponding button, a point will be added to his score and the next sequence will include two random values to remember. This process continues until the player misses a round. The score is shown using multiplexing of 2 digits on the 7-segment display, maxing out at a score of 99. The challenging part of this project includes implementing a non-fixed sequence sizes and creating a game that is difficult but still fun. The project is lightly based off of a “Simon Says” style memorization game, with four colors indicating which button must be pressed.

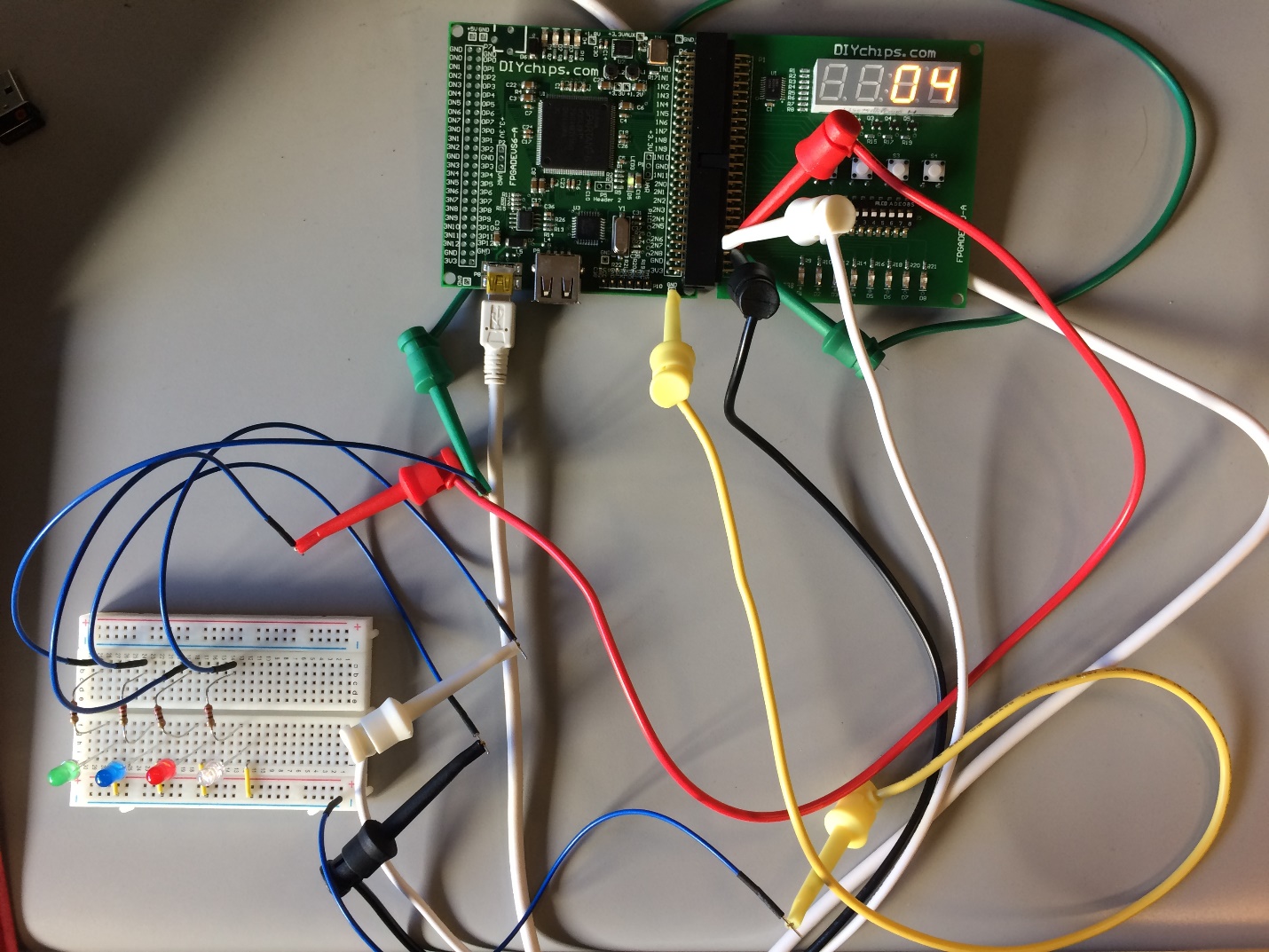
Hardware used:

* Xilinx Spartan-6 FPGA board
* 4 colored LED’s (green, blue, red, white)
* Miscellaneous wires and resistors

Software used:

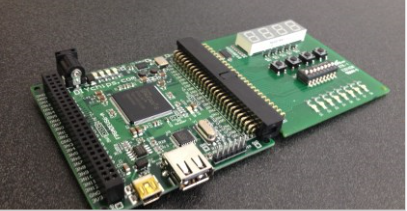
* Verilog HDL

**Project Picture:** *(initial state of decrementing with no user entry)*

****

**Hardware diagram:**

Pushbuttons – arranged in green, blue, red, white order for sequence entry



LED’s for sequence display

**Block diagram:**

Score incremented and displayed

Player enters correct sequence value (pushbutton)

New sequence generated

External LED display

Player enters incorrect or nonexistent sequence value (pushbutton)

Score decremented every second

**Module Description:**

board.ucf – used in all labs in this course.

Student\_Project.v – could have been broken up into separate modules; for simplicity’s sake all included under separate “always” blocks; will name these with comments and explain as such below:

* Clock1 and clock2 generation – define two clocks specified at 1 and 100 Hertz, respectively. The first is used for general-purposed project implementation block and the second is used for 7-segment display multiplexing.
* Score display – basic case statement giving representation of score on 7-segment display.
* Display multiplexing – based on clock2, this module multiplexes the two right-most 7-segment displays using variable “display”.
* I/O modeling – “always” block mapping “sequence” variable to LED’s and pushbuttons to “PBarray” variable for comparisons.
* Memorization implementation – using a two state FSM with “show” and “ready”, this block generates 8 unique rounds, each with their own sequence of LED values. It then sets the corresponding bit in the placeholder array to 1. After lighting up for just enough time for the user to see and evaluate (i.e. set to 1 second), it then lights up with additional sequence values depending on total sequence length. Finally, the FSM controls how score is allocated and if the player has answered with the correct value. Depending on the player’s performance, another round may start with a new sequence and or score may continue to be decremented (1 point every second). If the score reaches 0, the game ends and the player loses. If the score reaches the maximum (99), the player has won. Also of note is the fact that the player is initially given 9 points so that they have a small amount of time to memorize the first sequence while their score starts to be decremented. Score is allocated in ascending order. For example, the first value in each sequence is worth 1 point, the second 2 points, and so on and so forth. Lastly, the game can go up to a total of 8 rounds; in this final round there are 8 values to memorize.

**Source Code:** *(long because of brute force “round” FSM)*

module Student\_Project(input [3:0] IO\_PB, input [7:0] IO\_DSW, input M\_CLOCK,

output reg [7:0] IO\_SSEG, output reg [3:0] IO\_SSEGD, output [3:0] F\_LED, output IO\_SSEG\_COL, output reg [7:0] IO\_LED);

reg [31:0] CYCLE\_COUNT = 32'b00000001011111010111100001000000; // (50MHz / (1Hz) / 2) = 25,000,000 cycles (1 second)

reg [31:0] CycleCounter = 32'b00000000000000000000000000000000;

reg clock1 = 1'b0;

reg [31:0] CYCLE\_COUNT200Hz = 32'b00000000000000111101000010010000; // (50MHz / (50ms \* 2 = 100Hz) / 2) = 250,000 cycles

reg [31:0] CycleCounter2 = 32'b00000000000000000000000000000000;

reg clock2 = 1'b0;

integer display = 0;

// Sequence placeholders.

reg [7:0] sequence = 8'b00000000;

reg [3:0] sequenceHold = 4'b0000;

reg [3:0] sequenceHold1 = 4'b0000;

reg [3:0] sequenceHold2 = 4'b0000;

reg [3:0] sequenceHold3 = 4'b0000;

reg [3:0] sequenceHold4 = 4'b0000;

reg [3:0] sequenceHold5 = 4'b0000;

reg [3:0] sequenceHold6 = 4'b0000;

reg [3:0] sequenceHold7 = 4'b0000;

reg [3:0] score0 = 4'b0000;

reg [3:0] score1 = 4'b0000;

integer decrementFlag = 0;

reg [3:0] PBarray;

// For display.

reg [7:0] score0segs = 8'b11000000;

reg [7:0] score1segs = 8'b11000000;

assign F\_LED = 4'b0000;

assign IO\_SSEG\_COL = 1'b1;

always @(posedge M\_CLOCK) begin

if(CycleCounter <= CYCLE\_COUNT) begin

CycleCounter = CycleCounter + 1;

end

else begin

CycleCounter = 0;

clock1 = ~clock1;

end

end

always @(posedge M\_CLOCK) begin

if(CycleCounter2 <= CYCLE\_COUNT200Hz) begin

CycleCounter2 = CycleCounter2 + 1;

end

else begin

CycleCounter2 = 0;

clock2 = ~clock2;

end

end

always @\* begin

case(score0)

4'b0000: score0segs = 8'b11000000; //0

4'b0001: score0segs = 8'b11111001; //1

4'b0010: score0segs = 8'b10100100; //2

4'b0011: score0segs = 8'b10110000; //3

4'b0100: score0segs = 8'b10011001; //4

4'b0101: score0segs = 8'b10010010; //5

4'b0110: score0segs = 8'b10000010; //6

4'b0111: score0segs = 8'b11111000; //7

4'b1000: score0segs = 8'b10000000; //8

4'b1001: score0segs = 8'b10011000; //9

default: score0segs = 8'b11000000;

endcase

case(score1)

4'b0000: score1segs = 8'b11000000; //0

4'b0001: score1segs = 8'b11111001; //1

4'b0010: score1segs = 8'b10100100; //2

4'b0011: score1segs = 8'b10110000; //3

4'b0100: score1segs = 8'b10011001; //4

4'b0101: score1segs = 8'b10010010; //5

4'b0110: score1segs = 8'b10000010; //6

4'b0111: score1segs = 8'b11111000; //7

4'b1000: score1segs = 8'b10000000; //8

4'b1001: score1segs = 8'b10011000; //9

default: score1segs = 8'b11000000;

endcase

end

always @(posedge clock2) begin // 100Hz clock.

if(display == 1)

display = 0;

else

display = display + 1;

case(display)

0: begin

IO\_SSEGD[3:0] = 4'b0111;

IO\_SSEG = score0segs; // Given above.

end

1: begin

IO\_SSEGD[3:0] = 4'b1011;

IO\_SSEG = score1segs; // Given above.

end

default: begin

IO\_SSEGD[3:0] = 4'b1111;

IO\_SSEG = 8'b11001100;

end

endcase

end

always @\* begin

IO\_LED[7:0] = sequence;

{PBarray[0], PBarray[1], PBarray[2], PBarray[3]} = ~IO\_PB[3:0];

end

// Timing counters.

integer i = 0;

integer j = 0;

integer k = 0;

integer l = 0;

integer m = 0;

integer n = 0;

integer p = 0;

integer a = 0;

// State machine variables.

parameter SHOW = 0;

parameter READY = 1;

reg state = SHOW;

reg [2:0] round = 3'b000;

reg [2:0] order = 3'b000;

// Simon says placeholders.

reg [3:0] simonSays = 4'b0000;

reg [3:0] simonSays1 = 4'b0000;

reg [3:0] simonSays2 = 4'b0000;

reg [3:0] simonSays3 = 4'b0000;

reg [3:0] simonSays4 = 4'b0000;

reg [3:0] simonSays5 = 4'b0000;

reg [3:0] simonSays6 = 4'b0000;

reg [3:0] simonSays7 = 4'b0000;

// Memorization game.

always @(posedge clock1) begin

if(state == SHOW) begin // SHOW state.

//$srandom(3);

//color = $urandom\_range(3,0);

if(round == 3'b000) begin // Round 1.

simonSays[2] = 1'b1; // Set value.

sequence[7:4] = simonSays; // Set LED to show correct color. (ON)

sequenceHold = simonSays;

score0 = 4'b1001; // Give player 9 score (9 second countdown).

score1 = 4'b0000;

decrementFlag = 0;

state = READY;

end

else if(round == 3'b001) begin // Round 2.

simonSays[0] = 1'b1; // Set value.

simonSays1[3] = 1'b1; // Set value.

sequenceHold = simonSays;

sequenceHold1 = simonSays1;

sequence[7:4] = simonSays;

i = i + 1;

if(state == SHOW && i > 1) begin

i = 0;

sequence[7:4] = simonSays1;

state = READY;

end

end

else if(round == 3'b010) begin // Round 3.

simonSays[0] = 1'b1; // Set value.

simonSays1[1] = 1'b1; // Set value.

simonSays2[2] = 1'b1; // Set value.

sequenceHold = simonSays;

sequenceHold1 = simonSays1;

sequenceHold2 = simonSays2;

sequence[7:4] = simonSays;

i = i + 1;

if(state == SHOW && i > 1) begin

sequence[7:4] = simonSays1;

j = j + 1;

if(state == SHOW && j > 1) begin

i = 0;

j = 0;

sequence[7:4] = simonSays2;

state = READY;

end

end

end

else if(round == 3'b011) begin // Round 4.

simonSays[3] = 1'b1; // Set value.

simonSays1[2] = 1'b1; // Set value.

simonSays2[1] = 1'b1; // Set value.

simonSays3[0] = 1'b1; // Set value.

sequenceHold = simonSays;

sequenceHold1 = simonSays1;

sequenceHold2 = simonSays2;

sequenceHold3 = simonSays3;

sequence[7:4] = simonSays;

i = i + 1;

if(state == SHOW && i > 1) begin

sequence[7:4] = simonSays1;

j = j + 1;

if(state == SHOW && j > 1) begin

sequence[7:4] = simonSays2;

k = k + 1;

if(state == SHOW && k > 1) begin

i = 0;

j = 0;

k = 0;

sequence[7:4] = simonSays3;

state = READY;

end

end

end

end

else if(round == 3'b100) begin // Round 5.

simonSays[2] = 1'b1; // Set value.

simonSays1[3] = 1'b1; // Set value.

simonSays2[2] = 1'b1; // Set value.

simonSays3[3] = 1'b1; // Set value.

simonSays4[0] = 1'b1; // Set value.

sequenceHold = simonSays;

sequenceHold1 = simonSays1;

sequenceHold2 = simonSays2;

sequenceHold3 = simonSays3;

sequenceHold4 = simonSays4;

sequence[7:4] = simonSays;

i = i + 1;

if(state == SHOW && i > 1) begin

sequence[7:4] = simonSays1;

j = j + 1;

if(state == SHOW && j > 1) begin

sequence[7:4] = simonSays2;

k = k + 1;

if(state == SHOW && k > 1) begin

sequence[7:4] = simonSays3;

l = l + 1;

if(state == SHOW && l > 1) begin

i = 0;

j = 0;

k = 0;

l = 0;

sequence[7:4] = simonSays4;

state = READY;

end

end

end

end

end

else if(round == 3'b101) begin // Round 6.

simonSays[3] = 1'b1; // Set value.

simonSays1[3] = 1'b1; // Set value.

simonSays2[0] = 1'b1; // Set value.

simonSays3[1] = 1'b1; // Set value.

simonSays4[2] = 1'b1; // Set value.

simonSays5[3] = 1'b1; // Set value.

sequenceHold = simonSays;

sequenceHold1 = simonSays1;

sequenceHold2 = simonSays2;

sequenceHold3 = simonSays3;

sequenceHold4 = simonSays4;

sequenceHold5 = simonSays5;

sequence[7:4] = simonSays;

i = i + 1;

if(state == SHOW && i > 1) begin

sequence[7:4] = simonSays1;

j = j + 1;

if(state == SHOW && j > 1) begin

sequence[7:4] = simonSays2;

k = k + 1;

if(state == SHOW && k > 1) begin

sequence[7:4] = simonSays3;

l = l + 1;

if(state == SHOW && l > 1) begin

sequence[7:4] = simonSays4;

m = m + 1;

if(state == SHOW && m > 1) begin

i = 0;

j = 0;

k = 0;

l = 0;

m = 0;

sequence[7:4] = simonSays5;

state = READY;

end

end

end

end

end

end

else if(round == 3'b110) begin // Round 7.

simonSays[2] = 1'b1; // Set value.

simonSays1[1] = 1'b1; // Set value.

simonSays2[3] = 1'b1; // Set value.

simonSays3[0] = 1'b1; // Set value.

simonSays4[1] = 1'b1; // Set value.

simonSays5[2] = 1'b1; // Set value.

simonSays6[2] = 1'b1; // Set value.

sequenceHold = simonSays;

sequenceHold1 = simonSays1;

sequenceHold2 = simonSays2;

sequenceHold3 = simonSays3;

sequenceHold4 = simonSays4;

sequenceHold5 = simonSays5;

sequenceHold6 = simonSays6;

sequence[7:4] = simonSays;

i = i + 1;

if(state == SHOW && i > 1) begin

sequence[7:4] = simonSays1;

j = j + 1;

if(state == SHOW && j > 1) begin

sequence[7:4] = simonSays2;

k = k + 1;

if(state == SHOW && k > 1) begin

sequence[7:4] = simonSays3;

l = l + 1;

if(state == SHOW && l > 1) begin

sequence[7:4] = simonSays4;

m = m + 1;

if(state == SHOW && m > 1) begin

sequence[7:4] = simonSays5;

n = n + 1;

if(state == SHOW && n > 1) begin

i = 0;

j = 0;

k = 0;

l = 0;

m = 0;

n = 0;

sequence[7:4] = simonSays6;

state = READY;

end

end

end

end

end

end

end

else if(round == 3'b111) begin // Round 8.

simonSays[1] = 1'b1; // Set value.

simonSays1[0] = 1'b1; // Set value.

simonSays2[3] = 1'b1; // Set value.

simonSays3[2] = 1'b1; // Set value.

simonSays4[1] = 1'b1; // Set value.

simonSays5[3] = 1'b1; // Set value.

simonSays6[1] = 1'b1; // Set value.

simonSays7[2] = 1'b1; // Set value.

sequenceHold = simonSays;

sequenceHold1 = simonSays1;

sequenceHold2 = simonSays2;

sequenceHold3 = simonSays3;

sequenceHold4 = simonSays4;

sequenceHold5 = simonSays5;

sequenceHold6 = simonSays6;

sequenceHold7 = simonSays7;

sequence[7:4] = simonSays;

i = i + 1;

if(state == SHOW && i > 1) begin

sequence[7:4] = simonSays1;

j = j + 1;

if(state == SHOW && j > 1) begin

sequence[7:4] = simonSays2;

k = k + 1;

if(state == SHOW && k > 1) begin

sequence[7:4] = simonSays3;

l = l + 1;

if(state == SHOW && l > 1) begin

sequence[7:4] = simonSays4;

m = m + 1;

if(state == SHOW && m > 1) begin

sequence[7:4] = simonSays5;

n = n + 1;

if(state == SHOW && n > 1) begin

sequence[7:4] = simonSays6;

p = p + 1;

if(state == SHOW && p > 1) begin

i = 0;

j = 0;

k = 0;

l = 0;

m = 0;

n = 0;

p = 0;

sequence[7:4] = simonSays7;

state = READY;

end

end

end

end

end

end

end

end

a = 0;

end

// Turn off last LED.

a = a + 1;

if(state == READY && a > 1) begin

a = 0;

sequence[7:4] = 4'b0000;

end

if(state == READY) begin // READY state.

if(round == 3'b000) begin // Round 1.

if(sequenceHold == PBarray) begin

round = 3'b001;

state = SHOW;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b000;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

else if(round == 3'b001) begin // Round 2.

if(sequenceHold == PBarray && order == 3'b000) begin

round = 3'b001;

state = READY;

order = 3'b001;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold1 == PBarray && order == 3'b001) begin

round = 3'b010; // Advance.

state = SHOW;

order = 3'b000;

simonSays1 = 4'b0000;

sequenceHold1 = 4'b0000;

score0 = score0 + 2'b10;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b001;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

else if(round == 3'b010) begin // Round 3.

if(sequenceHold == PBarray && order == 3'b000) begin

round = 3'b010;

state = READY;

order = 3'b001;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold1 == PBarray && order == 3'b001) begin

round = 3'b010;

state = READY;

order = 3'b010;

simonSays1 = 4'b0000;

sequenceHold1 = 4'b0000;

score0 = score0 + 2'b10;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold2 == PBarray && order == 3'b010) begin

round = 3'b011;

state = SHOW;

order = 3'b000;

simonSays2 = 4'b0000;

sequenceHold2 = 4'b0000;

score0 = score0 + 2'b11;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b010;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

else if(round == 3'b011) begin // Round 4.

if(sequenceHold == PBarray && order == 3'b000) begin

round = 3'b011;

state = READY;

order = 3'b001;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold1 == PBarray && order == 3'b001) begin

round = 3'b011;

state = READY;

order = 3'b010;

simonSays1 = 4'b0000;

sequenceHold1 = 4'b0000;

score0 = score0 + 2'b10;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold2 == PBarray && order == 3'b010) begin

round = 3'b011;

state = READY;

order = 3'b011;

simonSays2 = 4'b0000;

sequenceHold2 = 4'b0000;

score0 = score0 + 2'b11;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold3 == PBarray && order == 3'b011) begin

round = 3'b100;

state = SHOW;

order = 3'b000;

simonSays3 = 4'b0000;

sequenceHold3 = 4'b0000;

score0 = score0 + 3'b100;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b011;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

else if(round == 3'b100) begin // Round 5.

if(sequenceHold == PBarray && order == 3'b000) begin

round = 3'b100;

state = READY;

order = 3'b001;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold1 == PBarray && order == 3'b001) begin

round = 3'b100;

state = READY;

order = 3'b010;

simonSays1 = 4'b0000;

sequenceHold1 = 4'b0000;

score0 = score0 + 2'b10;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold2 == PBarray && order == 3'b010) begin

round = 3'b100;

state = READY;

order = 3'b011;

simonSays2 = 4'b0000;

sequenceHold2 = 4'b0000;

score0 = score0 + 2'b11;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold3 == PBarray && order == 3'b011) begin

round = 3'b100;

state = READY;

order = 3'b100;

simonSays3 = 4'b0000;

sequenceHold3 = 4'b0000;

score0 = score0 + 3'b100;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold4 == PBarray && order == 3'b100) begin

round = 3'b101;

state = SHOW;

order = 3'b000;

simonSays4 = 4'b0000;

sequenceHold4 = 4'b0000;

score0 = score0 + 3'b101;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b100;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

else if(round == 3'b101) begin // Round 6.

if(sequenceHold == PBarray && order == 3'b000) begin

round = 3'b101;

state = READY;

order = 3'b001;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold1 == PBarray && order == 3'b001) begin

round = 3'b101;

state = READY;

order = 3'b010;

simonSays1 = 4'b0000;

sequenceHold1 = 4'b0000;

score0 = score0 + 2'b10;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold2 == PBarray && order == 3'b010) begin

round = 3'b101;

state = READY;

order = 3'b011;

simonSays2 = 4'b0000;

sequenceHold2 = 4'b0000;

score0 = score0 + 2'b11;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold3 == PBarray && order == 3'b011) begin

round = 3'b101;

state = READY;

order = 3'b100;

simonSays3 = 4'b0000;

sequenceHold3 = 4'b0000;

score0 = score0 + 3'b100;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold4 == PBarray && order == 3'b100) begin

round = 3'b101;

state = READY;

order = 3'b101;

simonSays4 = 4'b0000;

sequenceHold4 = 4'b0000;

score0 = score0 + 3'b101;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold5 == PBarray && order == 3'b101) begin

round = 3'b110;

state = SHOW;

order = 3'b000;

simonSays5 = 4'b0000;

sequenceHold5 = 4'b0000;

score0 = score0 + 3'b110;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b101;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

else if(round == 3'b110) begin // Round 7.

if(sequenceHold == PBarray && order == 3'b000) begin

round = 3'b110;

state = READY;

order = 3'b001;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold1 == PBarray && order == 3'b001) begin

round = 3'b110;

state = READY;

order = 3'b010;

simonSays1 = 4'b0000;

sequenceHold1 = 4'b0000;

score0 = score0 + 2'b10;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold2 == PBarray && order == 3'b010) begin

round = 3'b110;

state = READY;

order = 3'b011;

simonSays2 = 4'b0000;

sequenceHold2 = 4'b0000;

score0 = score0 + 2'b11;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold3 == PBarray && order == 3'b011) begin

round = 3'b110;

state = READY;

order = 3'b100;

simonSays3 = 4'b0000;

sequenceHold3 = 4'b0000;

score0 = score0 + 3'b100;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold4 == PBarray && order == 3'b100) begin

round = 3'b110;

state = READY;

order = 3'b101;

simonSays4 = 4'b0000;

sequenceHold4 = 4'b0000;

score0 = score0 + 3'b101;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold5 == PBarray && order == 3'b101) begin

round = 3'b110;

state = READY;

order = 3'b110;

simonSays5 = 4'b0000;

sequenceHold5 = 4'b0000;

score0 = score0 + 3'b110;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold6 == PBarray && order == 3'b110) begin

round = 3'b111;

state = SHOW;

order = 3'b000;

simonSays6 = 4'b0000;

sequenceHold6 = 4'b0000;

score0 = score0 + 3'b111;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b110;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

else if(round == 3'b111) begin // Round 8.

if(sequenceHold == PBarray && order == 3'b000) begin

round = 3'b111;

state = READY;

order = 3'b001;

simonSays = 4'b0000;

sequenceHold = 4'b0000;

score0 = score0 + 1'b1;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold1 == PBarray && order == 3'b001) begin

round = 3'b111;

state = READY;

order = 3'b010;

simonSays1 = 4'b0000;

sequenceHold1 = 4'b0000;

score0 = score0 + 2'b10;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold2 == PBarray && order == 3'b010) begin

round = 3'b111;

state = READY;

order = 3'b011;

simonSays2 = 4'b0000;

sequenceHold2 = 4'b0000;

score0 = score0 + 2'b11;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold3 == PBarray && order == 3'b011) begin

round = 3'b111;

state = READY;

order = 3'b100;

simonSays3 = 4'b0000;

sequenceHold3 = 4'b0000;

score0 = score0 + 3'b100;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold4 == PBarray && order == 3'b100) begin

round = 3'b111;

state = READY;

order = 3'b101;

simonSays4 = 4'b0000;

sequenceHold4 = 4'b0000;

score0 = score0 + 3'b101;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold5 == PBarray && order == 3'b101) begin

round = 3'b111;

state = READY;

order = 3'b110;

simonSays5 = 4'b0000;

sequenceHold5 = 4'b0000;

score0 = score0 + 3'b110;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold6 == PBarray && order == 3'b110) begin

round = 3'b111;

state = READY;

order = 3'b111;

simonSays6 = 4'b0000;

sequenceHold6 = 4'b0000;

score0 = score0 + 3'b111;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else if(sequenceHold7 == PBarray && order == 3'b111) begin

round = 3'b000; // Reset to first round.

state = SHOW;

order = 3'b000;

simonSays7 = 4'b0000;

sequenceHold7 = 4'b0000;

score0 = score0 + 4'b1000;

if(score0 > 4'b1001) begin

score0 = score0 - 4'b1010;

score1 = score1 + 1'b1;

end

end

else begin

round = 3'b111;

state = READY;

if((score0 != 4'b0000) || (score1 != 4'b0000)) begin // For decrementing score.

score0 = score0 - 4'b0001;

if(decrementFlag == 1) begin

score0 = 4'b1001;

score1 = score1 - 4'b0001;

decrementFlag = 0;

end

if(score0 == 0) begin

decrementFlag = 1;

end

end

else begin

decrementFlag = 0;

score0 = 1'b0;

end

end

end

end

end

endmodule

**User Instructions:**

Operation of the memory game is very simple. On startup, a single light will flash. The goal for the player is to memorize this color/placement and then press the corresponding push button on the board. The player should see a score tally on the 7-segment display.

As the sequence gets longer and harder, the player may have difficulty memorizing the full sequence. For every second that passes without a correct entry, a point will be deducted from the player’s score.

General rules –

* Order matters!
* The player starts with 9 points.
* A round is started when the entry for the previous round is completed.
* Points are allocated based on the complexity of the sequence (in ascending point distribution).
* If a score of 0 is reached, the player has lost.
* If a score of 99 is reached, the player has won.
* If all rounds are completed without a score of 99, the round will reset and the player will play until they fall to 0 points or reach 99 points.

*Note: Because of the nature of the pushbuttons, the player should hold down on each push button entry for approximately half of a second. If pressed too long or too short, it may result in anomalies in the function of the game.*

**Test and Validation Results:**

The project works as expected with the following caveats/disclosure –

* I initially wanted to use a random number generator for sequence values. Because this cannot be implemented in Verilog using a simple command, such as “srand” (and can only be implemented using a form of a linear shift-back register), I decided to give set values to my sequences. To make the project more difficult, I implemented the score decrementing so that the game is still complex.
* There are 8 different sequences, each with variable 4-bit length.
* I have kept the project in a single module is that is how I designed it initially and didn’t have the time to de-modularize it.
* The program is very long; I used a brute force method to get my FSM working for the rounds. I initially started with just two states, but it eventually became very long. I most likely could have done this using far less lines of code (i.e. “for” loops), but this works and I didn’t want to test breaking the project.
* I also added the additional functionality of creating the “Simon Says” game by wiring up 4 LED’s to the FPGA. This has no functional difference with the implementation of the program, but makes memorization easier and more fun. People like to memorize “Blue, green, white, red”, instead of memorizing relative positions of same colored lights.

**Lab Notes:**

My only lab notes include the list of my set sequence values for the program, which are helpful for testing and validation –

* Round 1: Blue
* Round 2: White, Green
* Round 3: White, Red, Blue
* Round 4: Green, Blue, Red, White
* Round 5: Blue, Green, Blue, Green, White
* Round 6: Green, Green, White, Red, Blue, Green
* Round 7: Blue, Red, Green, White, Red, Blue, Blue
* Round 8: Red, White, Green, Blue, Red, Green, Red, Blue