Project Progress Report

Memorization game using non-fixed memory size and colored LED’s matched with pushbuttons.

Issues faced:

* Need to figure out rand() algorithm in HDL.
* Need to construct variable sized array for dynamic memory size.

Things done so far:

* Acquired LED’s, resistors, and wires.
* Module for score multiplexing taken from digital clock lab.
* Wrote top-most module with pseudo-code for implementation of logic:

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

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

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

reg clock1 = 1'b0;

integer display = 0;

reg [3:0] pattern;

always @(posedge M\_CLOCK) begin

if(CycleCounter <= CYCLE\_COUNT) begin

CycleCounter = CycleCounter + 1;

end

else begin

CycleCounter = 0;

clock1 = ~clock1;

end

end

always @\* begin

case(hrs1)

4'b0000: hrs1segs = 8'b11000000; //0

4'b0001: hrs1segs = 8'b11111001; //1

4'b0010: hrs1segs = 8'b10100100; //2

4'b0011: hrs1segs = 8'b10110000; //3

4'b0100: hrs1segs = 8'b10011001; //4

4'b0101: hrs1segs = 8'b10010010; //5

4'b0110: hrs1segs = 8'b10000010; //6

4'b0111: hrs1segs = 8'b11111000; //7

4'b1000: hrs1segs = 8'b10000000; //8

4'b1001: hrs1segs = 8'b10011000; //9

default: hrs1segs = 8'b11000000;

endcase

case(hrs1)

4'b0000: hrs1segs = 8'b11000000; //0

4'b0001: hrs1segs = 8'b11111001; //1

4'b0010: hrs1segs = 8'b10100100; //2

4'b0011: hrs1segs = 8'b10110000; //3

4'b0100: hrs1segs = 8'b10011001; //4

4'b0101: hrs1segs = 8'b10010010; //5

4'b0110: hrs1segs = 8'b10000010; //6

4'b0111: hrs1segs = 8'b11111000; //7

4'b1000: hrs1segs = 8'b10000000; //8

4'b1001: hrs1segs = 8'b10011000; //9

default: hrs1segs = 8'b11000000;

endcase

end

always @(posedge clock1) begin // 100Hz clock.

if(display == 1)

display = 0;

else

display = display + 1;

case(display)

0: begin

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

IO\_SSEG = score0; // Given above.

end

1: begin

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

IO\_SSEG = score1; // Given above.

end

default: begin

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

IO\_SSEG = 8'b11001100;

end

endcase

end

always @\* begin

sequence = IO\_LED[7:4];

end

integer i;

integer length = 0;

**always @\* begin // Need to develop further.**

**for(i = 0; i <= length; i = i + 1) begin**

**if(sequence == ~IO\_PB) begin**

**score0 = score0 + 1;**

**if(score0 == 9) begin**

**score0 = 0;**

**score1 = score1 + 1;**

**end**

**// Show new random sequence.**

**end**

**else if(sequence != ~IO\_PB) begin**

**//Show new random sequence.**

**end**

**end**

**endmodule**