BILKENT UNIVERSITY

Computer Engineering

CS 223 DIGITAL DESIGN

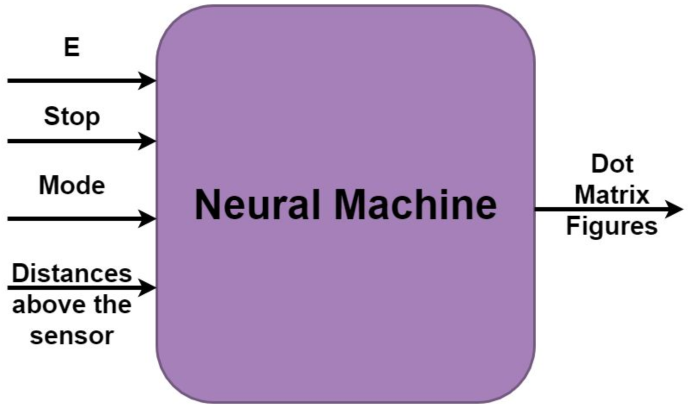
BRAIN:M

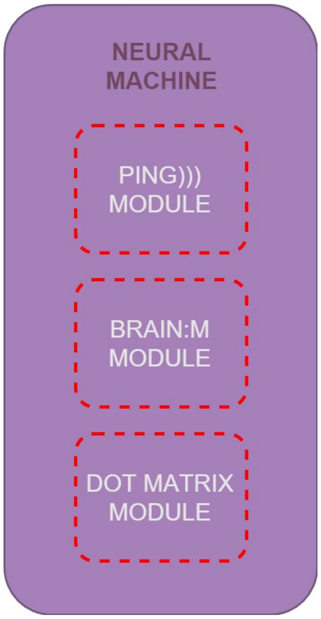
İmge GÖKALP 21402076 Section 01

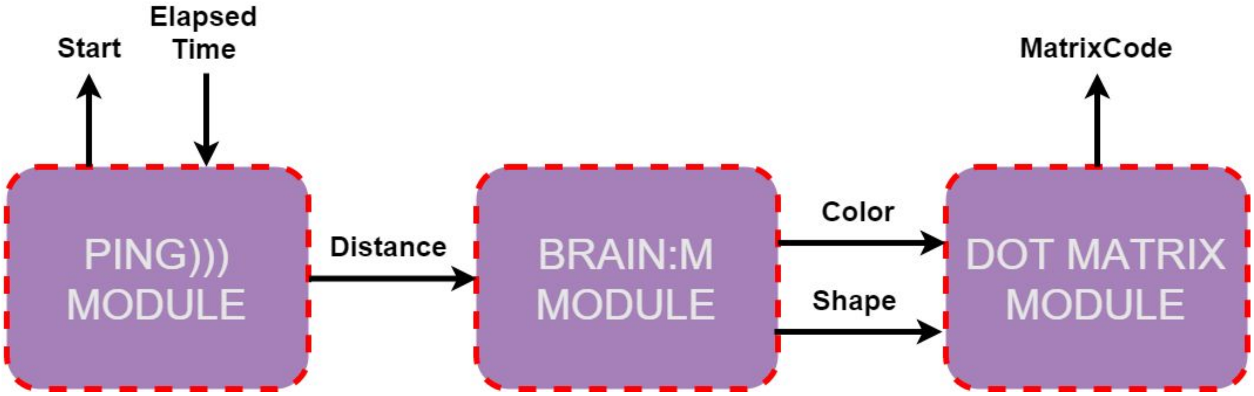
Mert INAN 21402020 Section 01

23/12/2015

General Block Diagram

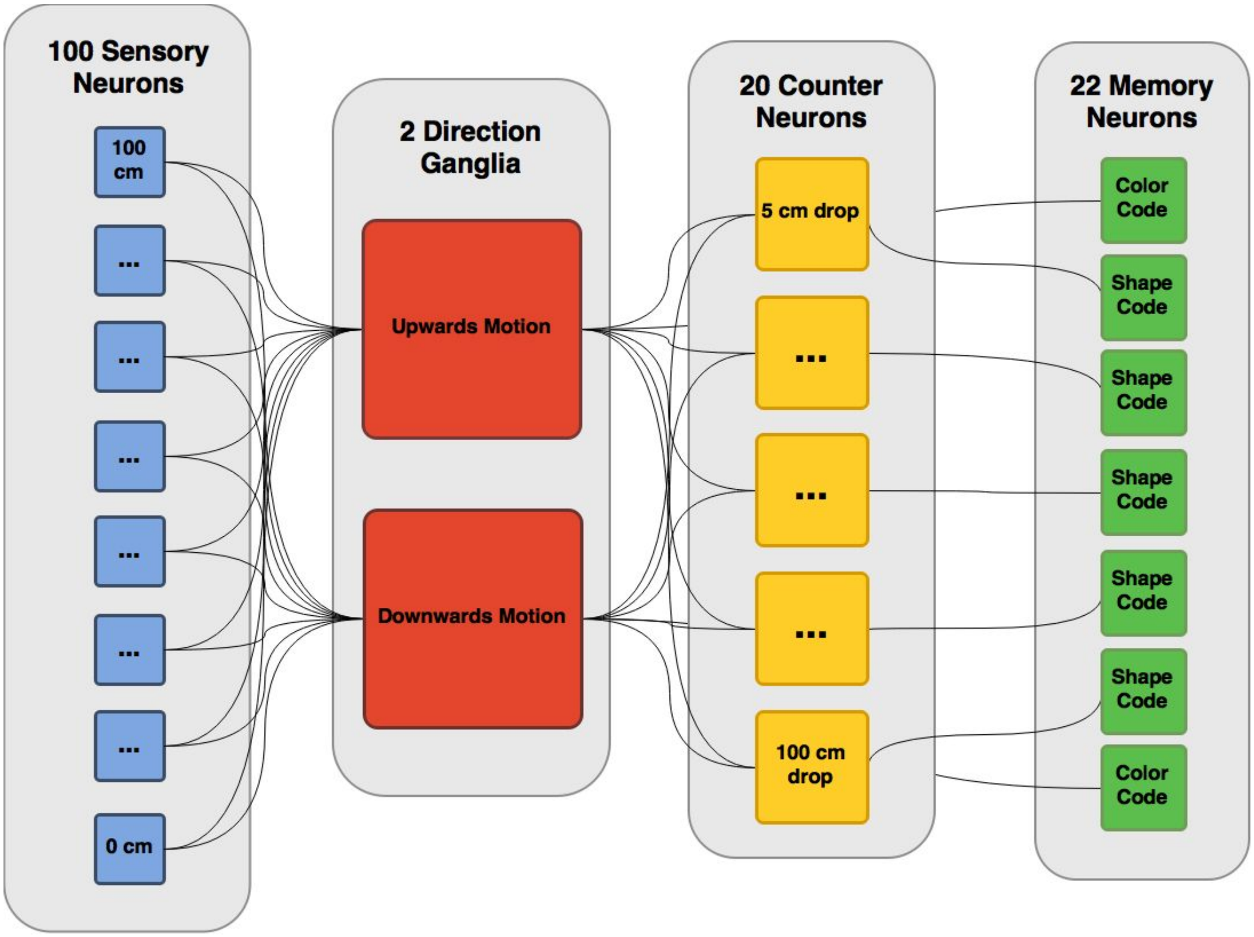




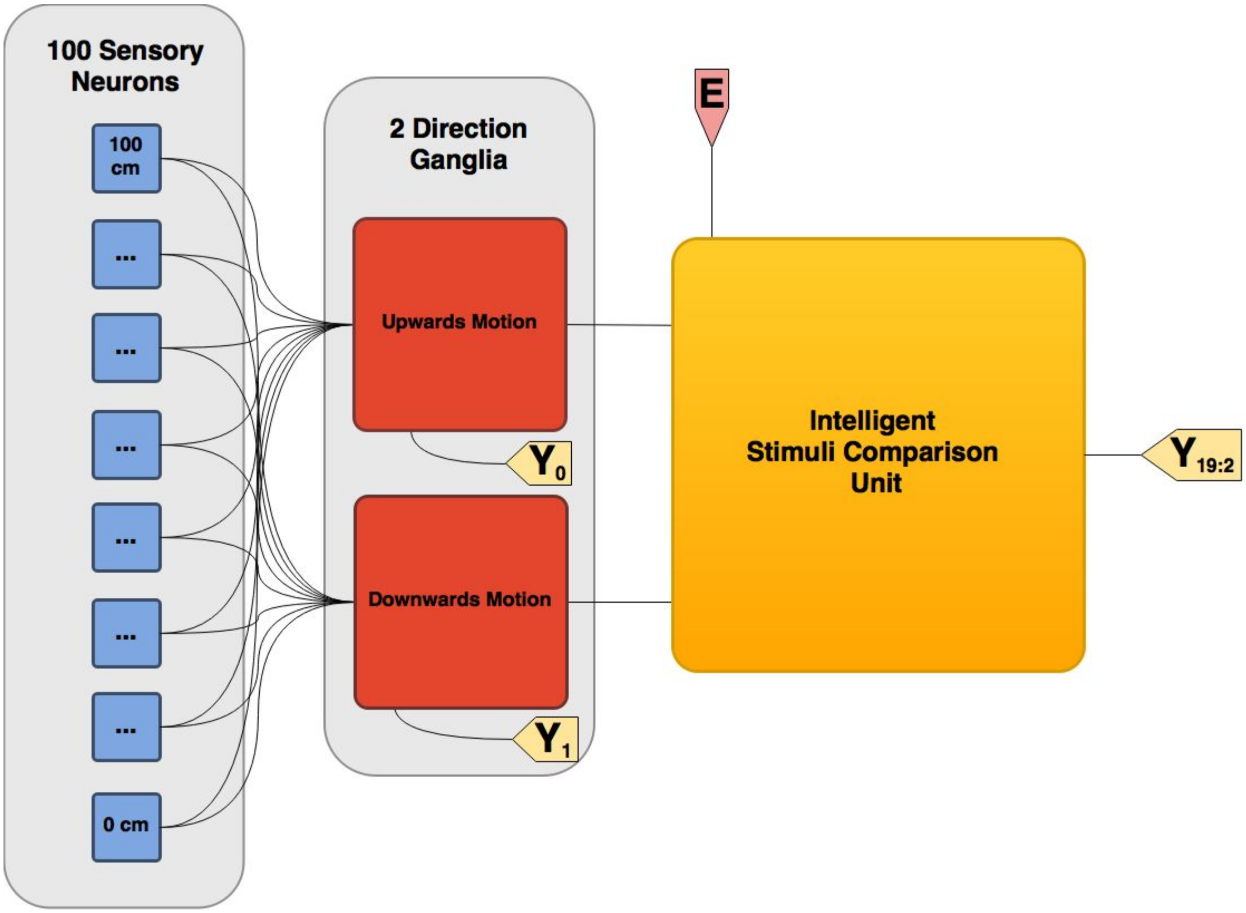


BRAIN:M Detailed Block Diagrams

Learning Mode



Demonstration Mode



Block Descriptions

PING))) Module

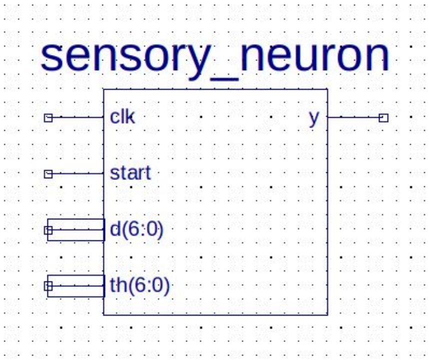
This block is the main part in which we gather the required distance information to work with in the BRAIN:M section. The timing constraints are rather dependent on the apparatus itself, because there are certain amounts of fixed waiting times in which the controller waits for the echo pulse, or tries to send a pulse, and checks whether there is no return signal. Mainly, this part sends a start signal to the sensor to start taking inputs by sending a pulse of high frequency sound, and listens to it. This PING))) controller will then get high input for the time being in which sound travels to and fro the object. This time length is divided by two to eliminate the first traveling time and then distance of the object is calculated using the elapsed time. Then this distance value is sent to the BRAIN:M module.

BRAIN:M Module

This section is the key part of the project as it involves the whole neural mechanism to learn from the distance of an object -whose data is gathered from the ultrasonic module- and output a shape with a certain color for the dot matrix. Because there are a lot of modules inside this module, there is a significant amount of delay from the input side to the output side. In simulations, the output delay was observed to be nearly 100 ms. This module contains five different modules in itself. There are a hundred sensory neuron modules that use the distance values gathered from the sensor up to 100 cm; and there are two direction ganglia, twenty interneurons, one memory neuron array and a stimuli comparison unit. All of these modules are wired to each other in this top-module using structural Verilog style.

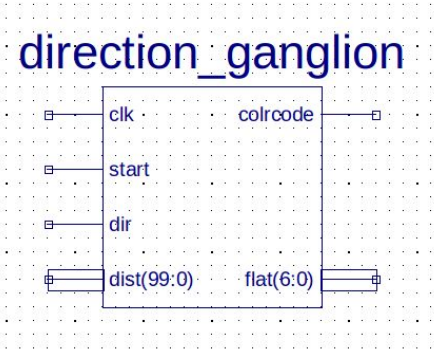
Sensory Neuron

This block of the system is a high level state machine which takes distance values from the sensor and outputs high if the distance is equal to its threshold value. The timing constraints are minimal, as there are not much of datapath elements and controller logic. Sensory neuron module functions according to its threshold value which is given as an input. Because this value and the distance value are multi-bit values, a simple finite state machine is not enough to recognize the distance and compare it with the threshold. Thus, an HLSM compares them using comparators in the datapath and starts this process when the start signal is high.



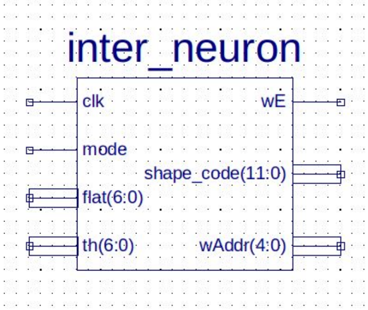
If the comparison from the datapath returns true, then the controller outputs high.

Direction Ganglion



Direction ganglia will wait for a low start signal while getting inputs from the sensory neuron and determine the change in the distance measured by the sensor. The timing constraints change depending on the timespan between the high and low start signals, yet the observed value after setting start signal to low is a delay of 35 ms in the simulator to output both the flattened distance and the color code. The main input to the direction ganglion is the one-hot data received collectively from hundred different sensory neurons. This creates a hundred bit input, and this input is repetitively analyzed at one-second intervals to spot the change in the direction. It stores the last two measured one-hot distance values in separate registers, and compares them with each other. If the values do not seem to change then it skips to the next set of data, if the second dataset is smaller than the first one, then it increments a counter inside it called descend counter; if it is the other way around then it increments the ascend counter. After the start signal is turned off, and there are no longer any distance values, then the ganglion outputs the counted value of ascension if the direction input is high, if not then it outputs the descend counter. It also gives a high output for color code if the motion is an ascend, and low if it is a descend. By doing this, it chooses a color for the dot matrix to show.

Counter Neuron/ Interneuron

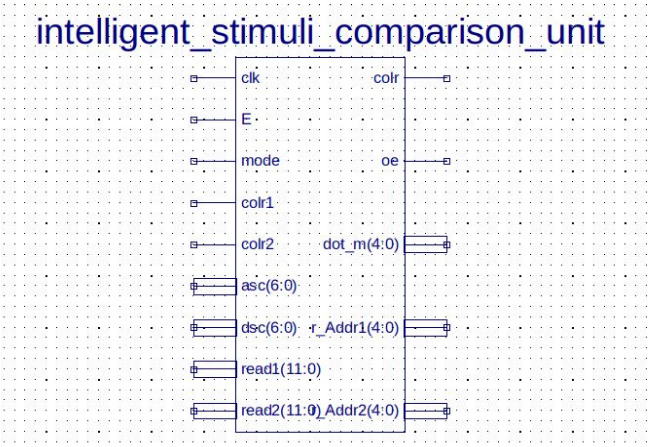


Interneuron is the main decision-maker for the shape that is related with the learned distance stimuli. The timing constraints are the same as the sensory neuron's. The main difference at this point of the neural level is that we now need to consider, whether the neural machine is in the learning mode or the demonstration mode. If it is in the demonstration mode then interneuron will be active rather than the intelligent stimuli-comparison unit. Interneuron takes the flattened distance value from the direction ganglia, because the direction of the movement is not necessary at this point, only the flattened distance from the ascending counter ganglion is used. It compares the flattened value with its threshold and according to that sends necessary address information to the memory neurons combined with concatenated data consisting of the flattened distance and a chosen shape code. It also sends a write enable. There will be twenty counter neurons in this module, thus it can only learn descents or ascents of multiples of five up to a hundred.

Memory Neuron

This module is implemented using a register file which holds twenty different twelve bit words in it that are written to it by the interneurons. Because there is no direct relation between the inputs and the outputs of the memory neuron array, the calculations and observations of timing constraints are irrelevant. The main important aspect of this module is that it is a very special register file with twenty write ports. It is necessary to use twenty write ports because all interneurons are going to write to the memory neuron array at the same time. However, there will be no overlapping write addresses as each address is linked with a unique value in the interneurons. This module also connects with the intelligent stimuli-comparison unit. It provides the values in the memory neurons via the two read ports.

Intelligent Stimuli Comparison Unit



This module compares the newly received distance values with the ones stored at the memory neuron array and tries to find the maximum -or minimum depending on E, the energy of the system- amount of times the distance in the memory neuron inside the new distance. The timing constraints are rather big for this module as it has a lot of registers and a lot of communication going on with the register file. This is the most complex section of the whole project, as it constitutes the core part of the BRAIN:M module. First of all, it starts to search for all the values in the memory neuron array from the end or from the beginning depending on the energy input E. After that, it looks for the ascending distance first and then the descending distance, in both cases of energy. It then, tries to subtract the highest amount from the ascending distance and substitutes the remainder of the operation into the whole search process again. By doing this it repetitively finds the maximum amount of occurrences of the memory distance inside the shown distance, and outputs that many times the shape encoded with that distance value. In the low energy case, it starts searching from the end of the array and it does not wait for the whole remainders to get smaller than the memory distances but it outputs the shape every time that it finds occurrences of the memory distance. At the end, this module outputs the color, and shape.

Dot Matrix Module

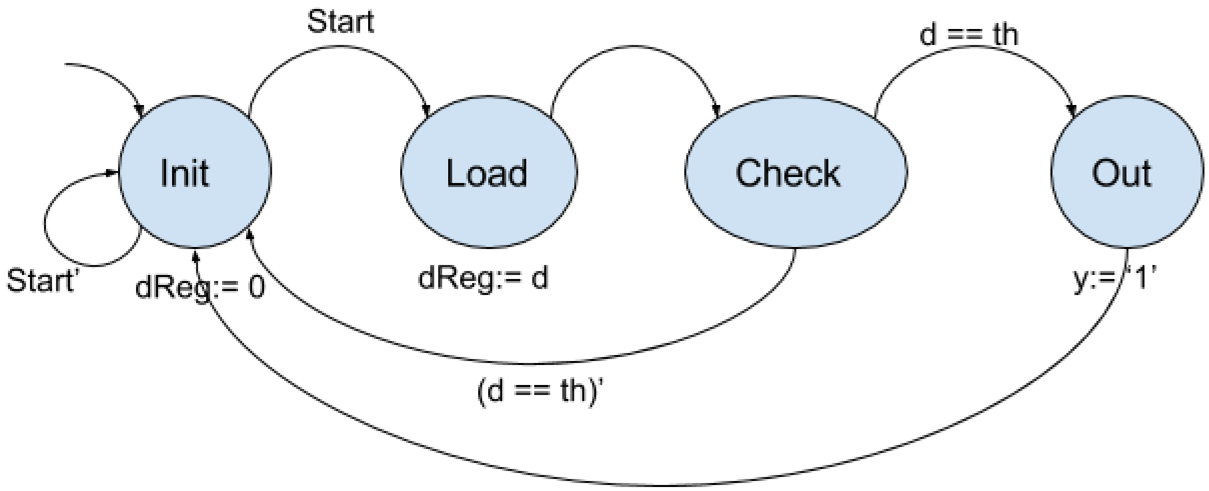
This is the module in which the outputs of BRAIN:M are guided to the dot matrix device. The time constraints are again minimal, yet there are counters and several other clocks that go into the dot matrix itself, which makes the timing of all the clocks important and synchronized. This is the controller of the dot matrix module found on the Beti boards, it takes the color code from the memory neurons of direction ganglia and the shape code from the memory neurons of counter neurons as inputs and combines them to produce the desired result which is then translated into the required style of code by the dot matrix device in the learning mode. In the demonstration mode, the inputs are taken from the intelligent stimuli-comparison unit.

References

Dot matrix and ultrasound controller modules are done by converting and modifying the VHDL files found on the demo CD of the Beti board. They are translated into Verilog and their input parameters are changed to suit our other modules' needs.

Appendix A: Verilog & HLSM

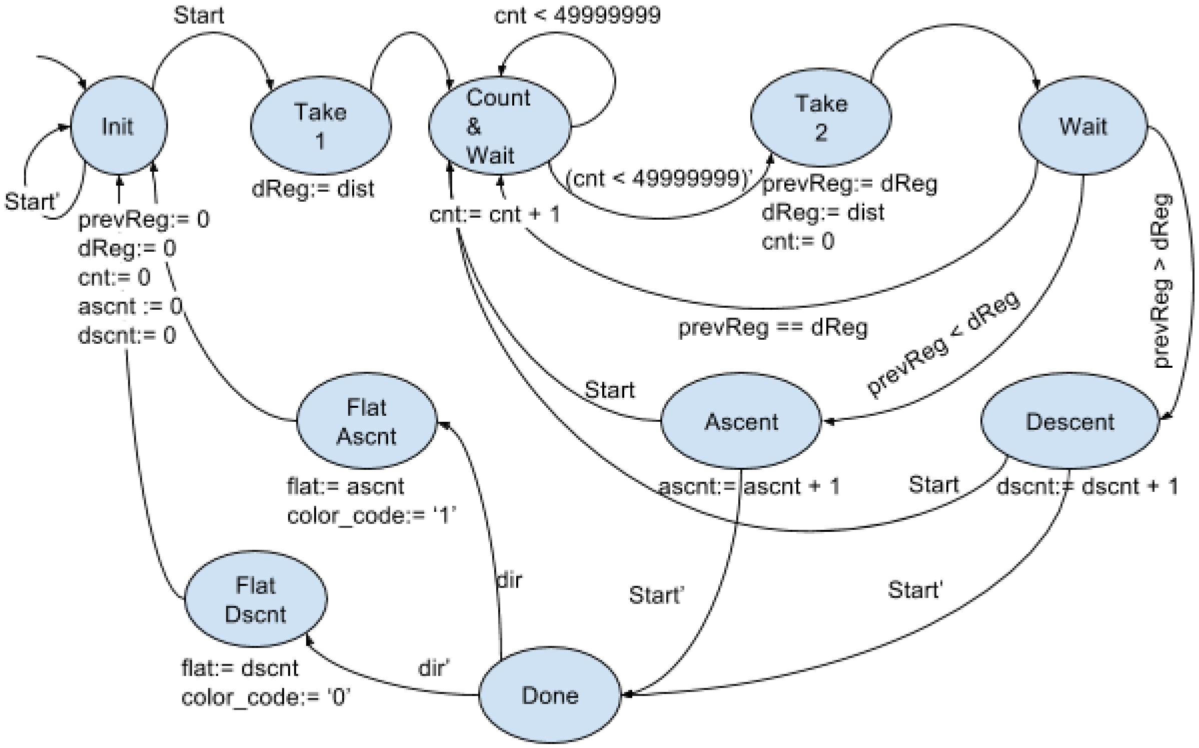
Sensory Neuron



```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Student: Mert İnan
//
// Create Date:    08:45:20 12/16/2015
// Module Name:    sensory_neuron
// Project Name:      BRAIN:M
// Target Devices: FPGA Board
// Description: This simple High Level State Machine is the smallest element of
//                  a brain that we instantiate in the top module. Its properties and
//                  actions are very similar to that of real sensory neurons found in
//                  mammalian brain. The sensory input for this neuron is the distance
//                  value measured by the PING))) Ultrasound module. It outputs high
//                  if the distance value is the same as its threshold value, which is
//                  also another input and is instantiated in the top module.
// Revision 0.01 - File Created
// Additional Comments: There will be a hundred of this sensory neuron in the
//                                  finished design. Instantiating a hundred neurons would
//                                  take a lot of space.
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
module sensory_neuron( clk, d, th, start, y);
	input clk, start;
	input [6:0] d, th;
	output reg y;
	reg [1:0] state, nextstate;
	reg [6:0] dReg, dNext;
```

```
    //Sequential Logic
    always @(posedge clk)
        begin
            state <= nextstate;
            dReg <= dNext;
        end

    //Combinational Logic
    always @( state or dReg or start or d or th)
        case (state)
            2'b00:                                          //Init State
                begin
                dNext = 7'b0;
                nextstate = (start)? 2'b01: 2'b00;
                y = 1'b0;
                end
            2'b01:                                          //Load State
                begin
                dNext = d;
                nextstate = 2'b10;
                y = 1'b0;
                end
            2'b10:                                          //Check State
                if (dReg == th)
                    nextstate = 2'b11;
                else
                    nextstate = 2'b00;
            2'b11:                                          //Out State
                begin
                y = 1'b1;
                nextstate = 2'b00;
                end
            default:
                begin
                y = 1'b0;
                dNext = 7'b0;
                nextstate = 2'b00;
                end
        endcase
endmodule
```

Direction Ganglion



```
//////////////////////////////////////////////////////////////////////////////////
// Student: Mert İnan
//
// Create Date:    16:31:40 12/19/2015
// Module Name:    direction_ganglion
// Project Name:      BRAIN:M
// Target Devices: Basys2 FPGA Board
//
// Description: This is the direction recognizing neural code that is used in the
//                    BRAIN:M module. It takes inputs from 100 sensory neurons and
//                    flattens that sensory stimuli array into a number and outputs it.
//
// Revision 0.01 - File Created
//////////////////////////////////////////////////////////////////////////////////
module direction_ganglion(clk, dist, start, dir, colrcode, flat);
        input clk, start, dir;
        input [99:0] dist;
        output reg colrcode;
        output reg [6:0] flat;
        reg [3:0] state, nextstate;
        reg [25:0] cnt, cntNext;
        reg [6:0] ascnt, ascntNext, dscnt, dscntNext;
        reg [99:0] dReg, dRegNext, prevReg, prevRegNext;
```

```
//Sequential Logic
always @(posedge clk) begin
        state <= nextstate;
        cnt <= cntNext;
        ascnt <= ascntNext;
        dscnt <= dscntNext;
        dReg <= dRegNext;
        prevReg <= prevRegNext;
end

//Combinational Logic
always @(*)
        case(state)
                4'b0000: begin
                        cntNext = 26'b0;
                        ascntNext = 7'b0;
                        dscntNext = 7'b0;
                        dRegNext = 100'b0;
                        prevRegNext = 100'b0;
                        nextstate= (start)? 4'b0001:4'b0000;
                end

                4'b0001: begin
                        dRegNext = dist;
                        nextstate = 4'b0010;
                end

                4'b0010: begin
                        cntNext = cnt + 1;
                        if (cnt < 26'b10111110101111000001111111)
                                nextstate = 4'b0010;
                        else
                                nextstate = 4'b0011;
                end

                4'b0011: begin
                        prevRegNext = dReg;
                        dRegNext = dist;
                        cntNext = 26'b0;
                        nextstate = 4'b0100;
                end

                4'b0100: begin
                        if  (prevReg > dReg)
                                nextstate = 4'b0101;
                else if (prevReg < dReg)
                    nextstate = 4'b0110;
                else
                    nextstate = 4'b0010;
            end

            4'b0101: begin
                dscntNext = dscnt + 1;
                nextstate = (start)? 4'b0010:4'b0111;
            end

            4'b0110: begin
                ascntNext = ascnt + 1;
                nextstate = (start)? 4'b0010:4'b0111;
            end

            4'b0111:
                nextstate = (dir)? 4'b1000:4'b1001;

            4'b1000: begin
                flat = ascnt;
                colrcode = 1'b1;
                nextstate = 4'b0000;
            end

            4'b1001: begin
                flat = dscnt;
                colrcode = 1'b0;
                nextstate = 4'b0000;
            end

            default: begin
                nextstate = 4'b0000;
                cntNext = 26'b0;
                ascntNext = 7'b0;
                dscntNext = 7'b0;
                dRegNext = 100'b0;
                prevRegNext = 100'b0;
                flat = 7'b0;
                colrcode = 1'b0;
            end
        endcase
endmodule
```

Interneuron

```
//////////////////////////////////////////////////////////////////////////////////
// Student: Mert İnan
//
// Create Date:    13:20:37 12/20/2015
// Module Name:    inter_neuron
// Project Name:      BRAIN:M
// Target Devices: FBGA Board
// Description: This is the module for the interneuron which is responsible of
//                    flow of information from the direction ganglia and the memory
//                    neuron array.
// Revision 0.01 - File Created
//////////////////////////////////////////////////////////////////////////////////

module inter_neuron(clk, flat, th, mode, shape_code, wAddr, wE);
    input clk, mode;
    input [6:0] th, flat;
    output reg wE;
    output reg [11:0] shape_code;
    output reg [4:0] wAddr;
    reg [4:0] state, nextstate, wAddrNext;
    reg [6:0] fReg, fRegNext;
    reg [11:0] shape_codeNext;

    //Sequential Logic
    always @(posedge clk) begin
        state <= nextstate;
        fReg <= fRegNext;
        wAddr <= wAddrNext;
        shape_code <= shape_codeNext;
    end

    //Combinational Logic
    always @(*)
        case(state)
        5'b00000: begin
            fRegNext = 7'b0;
            nextstate = (mode)? 5'b00000:5'b00001;
            wE = 1'b0;
        end

        5'b00001: begin
            fRegNext = flat;
            nextstate = 5'b00010;
            wE = 1'b0;
        end

        5'b00010:
            if (fReg == th)
                nextstate = 5'b00011;
            else
                nextstate = 5'b00000;
        5'b00011:
            case(th)
                7'b1100100: nextstate = 5'b00100;
                7'b1011111: nextstate = 5'b00101;
                7'b1011010: nextstate = 5'b00110;
                7'b1010101: nextstate = 5'b00111;
                7'b1010000: nextstate = 5'b01000;
                7'b1001011: nextstate = 5'b01001;
                7'b1000110: nextstate = 5'b01010;
                7'b1000001: nextstate = 5'b01011;
                7'b0111100: nextstate = 5'b01100;
                7'b0110111: nextstate = 5'b01101;
                7'b0110010: nextstate = 5'b01110;
                7'b0101101: nextstate = 5'b01111;
                7'b0101000: nextstate = 5'b10000;
                7'b0100011: nextstate = 5'b10001;
                7'b0011110: nextstate = 5'b10010;
                7'b0011001: nextstate = 5'b10011;
                7'b0010100: nextstate = 5'b10100;
                7'b0001111: nextstate = 5'b10101;
                7'b0001010: nextstate = 5'b10110;
                7'b0000101: nextstate = 5'b10111;
                default: nextstate = 5'b00011;
            endcase
//All states below here assign a new data to a specific address in the memory
//array according to the threshold.

        5'b00100: begin
            shape_codeNext = {fReg, 5'b00000};
            wAddrNext = 5'b00000;
            wE = 1'b1;
            nextstate = 5'b00011;
        end

        5'b00101: begin
            shape_codeNext = {fReg, 5'b00001};
            wAddrNext = 5'b00001;
```

```
    wE = 1'b1;
    nextstate = 5'b00011;
end

5'b00110: begin
    shape_codeNext = {fReg, 5'b00010};
    wAddrNext = 5'b00010;
    wE = 1'b1;
    nextstate = 5'b00011;
end

5'b00111: begin
    shape_codeNext = {fReg, 5'b00011};
    wAddrNext = 5'b00011;
    wE = 1'b1;
    nextstate = 5'b00011;
end

5'b01000: begin
    shape_codeNext = {fReg, 5'b00100};
    wAddrNext = 5'b00100;
    wE = 1'b1;
    nextstate = 5'b00011;
end

5'b01001: begin
    shape_codeNext = {fReg, 5'b00101};
    wAddrNext = 5'b00101;
    wE = 1'b1;
    nextstate = 5'b00011;
end

5'b01010: begin
    shape_codeNext = {fReg, 5'b00110};
    wAddrNext = 5'b00110;
    wE = 1'b1;
    nextstate = 5'b00011;
end

5'b01011: begin
    shape_codeNext = {fReg, 5'b00111};
    wAddrNext = 5'b00111;
    wE = 1'b1;
    nextstate = 5'b00011;
end
```

```
5'b01100: begin
        shape_codeNext = {fReg, 5'b01000};
        wAddrNext = 5'b01000;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b01101: begin
        shape_codeNext = {fReg, 5'b01001};
        wAddrNext = 5'b01001;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b01110: begin
        shape_codeNext = {fReg, 5'b01010};
        wAddrNext = 5'b01010;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b01111: begin
        shape_codeNext = {fReg, 5'b01011};
        wAddrNext = 5'b01011;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10000:
begin
        shape_codeNext = {fReg, 5'b01100};
        wAddrNext = 5'b01100;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10001: begin
        shape_codeNext = {fReg, 5'b01101};
        wAddrNext = 5'b01101;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10010: begin
```

```
        shape_codeNext = {fReg, 5'b01110};
        wAddrNext = 5'b01110;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10011: begin
        shape_codeNext = {fReg, 5'b01111};
        wAddrNext = 5'b01111;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10100: begin
        shape_codeNext = {fReg, 5'b10000};
        wAddrNext = 5'b10000;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10101: begin
        shape_codeNext = {fReg, 5'b10001};
        wAddrNext = 5'b10001;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10110: begin
        shape_codeNext = {fReg, 5'b10010};
        wAddrNext = 5'b10010;
        wE = 1'b1;
        nextstate = 5'b00011;
end

5'b10111:
begin
        shape_codeNext = {fReg, 5'b10011};
        wAddrNext = 5'b10011;
        wE = 1'b1;
        nextstate = 5'b00011;
end

default: begin
        nextstate = 5'b00000;
        shape_codeNext = shape_code;
```

```
            wAddrNext = wAddr;
            wE = 1'b0;
            fRegNext = fReg;
        end
    endcase
endmodule
```

Memory Neuron

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Student: Mert İnan
// Create Date:    14:40:55 12/20/2015
// Module Name:    memory_neuron
// Project Name:     BRAIN:M
// Target Devices:   FPGA Board
// Description:  This module is actually a cluster of neurons which hold specific
//                    memory of 12 bits, which is a concatenation of 7 bits of distance
//                    value and 5 bits of shape encoding values. This is a multiple read,
//                    multiple write register file. It has 20 write ports and 2 read ports.
//                    It gets its write data from the interneuron, and outputs from read
//                    ports to the intelligent stimuli comparison unit.
// Revision 0.01 - File Created
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
module memory_neuron(clk, WE1, WE2, WE3, WE4, WE5,
                                      WE6, WE7, WE8, WE9, WE10, WE11,
                                      WE12, WE13, WE14, WE15, WE16, WE17,
                                      WE18, WE19, WE20, wAddr1, wAddr2, wAddr3,
                                      wAddr4, wAddr5, wAddr6, wAddr7, wAddr8,
                                      wAddr9, wAddr10, wAddr11, wAddr12, wAddr13,
                                      wAddr14, wAddr15, wAddr16, wAddr17, wAddr18,
                                      wAddr19, wAddr20, wData1, wData2, wData3, wData4,
                                      wData5, wData6, wData7, wData8, wData9, wData10,
                                      wData11, wData12, wData13, wData14, wData15,
                                      wData16,  wData17, wData18, wData19, wData20,
                                      rAddr1, rAddr2, rData1, rData2);

        input clk, WE1, WE2, WE3, WE4, WE5,
                    WE6, WE7, WE8, WE9, WE10, WE11,
                    WE12, WE13, WE14, WE15, WE16, WE17,
                    WE18, WE19, WE20;

        input [4:0] wAddr1, wAddr2, wAddr3, wAddr4, wAddr5,
                    wAddr6, wAddr7, wAddr8, wAddr9, wAddr10,
                    wAddr11, wAddr12, wAddr13, wAddr14, wAddr15,
                    wAddr16, wAddr17, wAddr18, wAddr19, wAddr20,
                    rAddr1, rAddr2;

        input [11:0] wData1, wData2, wData3, wData4, wData5, wData6,
                    wData7, wData8, wData9, wData10, wData11, wData12,
                    wData13, wData14, wData15, wData16, wData17, wData18,
                    wData19, wData20;
```

```
output [11:0] rData1, rData2;

reg [11:0] mem[31:0];

always @(posedge clk)
        if (WE1)
                mem[wAddr1] <= wData1;
        else if (WE2)
                mem[wAddr2] <= wData2;
        else if (WE3)
                mem[wAddr3] <= wData3;
        else if (WE4)
                mem[wAddr4] <= wData4;
        else if (WE5)
                mem[wAddr5] <= wData5;
        else if (WE6)
                mem[wAddr6] <= wData6;
        else if (WE7)
                mem[wAddr7] <= wData7;
        else if (WE8)
                mem[wAddr8] <= wData8;
        else if (WE9)
                mem[wAddr9] <= wData9;
        else if (WE10)
                mem[wAddr10] <= wData10;
        else if (WE11)
                mem[wAddr11] <= wData11;
        else if (WE12)
                mem[wAddr12] <= wData12;
        else if (WE13)
                mem[wAddr13] <= wData13;
        else if (WE14)
                mem[wAddr14] <= wData14;
        else if (WE15)
                mem[wAddr15] <= wData15;
        else if (WE16)
                mem[wAddr16] <= wData16;
        else if (WE17)
                mem[wAddr17] <= wData17;
        else if (WE18)
                mem[wAddr18] <= wData18;
        else if (WE19)
                mem[wAddr19] <= wData19;
        else if (WE20)
```

```
            mem[wAddr20] <= wData20;
        else
            mem[wAddr20 + 1] <= wData1;

    assign rData1 = mem[rAddr1];
    assign rData2 = mem[rAddr2];
endmodule
```

Intelligent Stimuli-Comparison Unit

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Student: Mert İNAN
//
// Create Date:    18:06:42 12/19/2015
// Module Name:    intelligent_stimuli_comparison_unit
// Project Name:     BRAIN:M
// Target Devices:   FPGA Board
// Description: This is the main module that compares the already learnt values
//                  and the new stimuli using the shape_code values in the memory
//                  array. It outputs the appropriate colour and the shape to the dot
//                  matrix.
// Revision 0.01 - File Created
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module intelligent_stimuli_comparison_unit(clk, E, asc, dsc, mode, read1, read2,
                                              colr1, colr2, colr, oe, dot_m, r_Addr1, r_Addr2);
        input clk, E, mode, colr1, colr2;
        input [6:0] asc, dsc;
        input [11:0] read1, read2;
        output reg colr, oe;
        output reg [4:0] dot_m, r_Addr1, r_Addr2;

        reg [4:0] state, nextstate,  r_Addr1Next,  r_Addr2Next;
        reg [7:0] ascReg, ascRegNext, dscReg, dscRegNext,
                    remReg1, remReg1Next, remReg2, remReg2Next;
        reg [27:0] cnt, cntNext;

        //Sequential Logic
        always @(posedge clk)
        begin
                state <= nextstate;
                ascReg <= ascRegNext;
                dscReg <= dscRegNext;
                remReg1 <= remReg1Next;
                remReg2 <= remReg2Next;
                cnt <= cntNext;
                r_Addr1 <= r_Addr1Next;
                r_Addr2 <= r_Addr2Next;
        end

        //Combinational Logic
        always @(*)
                case(state)
```

```
5'b00000: //Initialization state
begin
    ascRegNext = 8'b0;
    dscRegNext = 8'b0;
    cntNext = 28'b0;
    remReg1Next = 8'b0;
    remReg2Next = 8'b0;
    nextstate = (mode)? 5'b00001: 5'b00000;
end

5'b00001:
begin //Load state
    ascRegNext = {1'b0, asc[6:0]}; //Concatenation for zero
    dscRegNext = {1'b0, dsc[6:0]}; //extension.
    nextstate = (E)? 5'b00010:5'b00011;
end

5'b00010:
begin //Read High state
    r_Addr1Next = 5'b0;
    r_Addr2Next = 5'b0;
    nextstate = 5'b00100;
end

5'b00011: //Read Low state
begin
    r_Addr1Next = 5'b10100;
    r_Addr2Next = 5'b10100;
    nextstate = 5'b10001;
end

5'b00100: //Remainder H state
begin
    remReg1Next = ascReg - {1'b0, read1[11:5]};
    remReg2Next = dscReg - {1'b0, read2[11:5]};
    if ( !(r_Addr1 < 5'b10100) && !(r_Addr2 < 5'b10100))
        nextstate = 5'b00000;
    else if ( !(r_Addr1 < 5'b10100) && (r_Addr2 < 5'b10100))
        nextstate = 5'b01011;
    else if (r_Addr1 < 5'b10100)
        nextstate = 5'b00101;
    else
     nextstate = 5'b00100;
end
```

```
5'b00101: //Wait High 1 state
if (remReg1[7] == 1)
        nextstate = 5'b00110;
else if (remReg1 == 8'b00000000)
        nextstate = 5'b01000;
else if (remReg1 > 8'b00000000)
        nextstate = 5'b01001;
else
        nextstate = 5'b00101;

5'b00110: // < High 1 state
begin
        r_Addr1Next = r_Addr1 + 1;
        nextstate = 5'b00111;
end

5'b00111:
if (r_Addr1 < 5'b10100)
        nextstate = 5'b00101;
else if (!(r_Addr1 < 5'b10100))
        nextstate = 5'b01011;
else
        nextstate = 5'b00111;

5'b01000: // = High 1 state
begin
        colr = colr1;
        dot_m = read1[4:0];
        cntNext = cnt + 1;
        oe = 1'b1;
        if (cnt < 28'b1000111100001101000101111111)
                nextstate = 5'b01000;
        else if (!(cnt < 28'b1000111100001101000101111111))
                nextstate = 5'b01010;
end

5'b01001: // > High 1 state
begin
        ascRegNext = remReg1;
        nextstate = 5'b00010;
end

5'b01010:
begin
        oe = 1'b0;
```

```
    cntNext = 28'b0;
    if (r_Addr1 < 5'b10100)
        nextstate = 5'b00101;
    else if (!(r_Addr1 < 5'b10100))
        nextstate = 5'b01011;
    else
        nextstate = 5'b01010;
end

5'b01011: //Wait High 2 state
if (remReg2[7] == 1)
    nextstate = 5'b01100;
else if (remReg2 == 8'b00000000)
    nextstate = 5'b01110;
else if (remReg2 > 8'b00000000)
    nextstate = 5'b01111;
else
    nextstate = 5'b01011;

5'b01100: //< High 2 state
begin
    r_Addr2Next = r_Addr2 + 1;
    nextstate = 5'b01101;
end

5'b01101:
if (r_Addr2 < 5'b10100)
    nextstate = 5'b01011;
else if (!(r_Addr2 < 5'b10100))
    nextstate = 5'b00000;
else
    nextstate = 5'b01101;

5'b01110: // = High 2 state
begin
    colr = colr2;
    dot_m = read2[4:0];
    cntNext = cnt + 1;
    oe = 1'b1;
    if (cnt < 28'b1000111100001101000101111111)
        nextstate = 5'b01110;
    else if (!(cnt < 28'b1000111100001101000101111111))
        nextstate = 5'b10000;
    else
        nextstate = 5'b01110;
```

```
end

5'b01111: // > High 2 state
begin
        dscRegNext = remReg2;
        r_Addr2Next = 5'b00000;
        nextstate = 5'b00100;
end

5'b10000:
begin
        oe = 1'b0;
        cntNext = 28'b0;
        if (r_Addr2 < 5'b10100)
                nextstate = 5'b01011;
        else if (!(r_Addr2 < 5'b10100))
                nextstate = 5'b00000;
        else
                nextstate = 5'b10000;
end

5'b10001: //Remainder Low state
begin
        remReg1Next = ascReg - {1'b0, read1[11:5]};
        remReg2Next = dscReg - {1'b0, read2[11:5]};
        cntNext = 28'b0;
        oe = 1'b0;
        if ((r_Addr1 == 5'b00000) && (r_Addr2 == 5'b00000))
                nextstate = 5'b00000;
        else if ((r_Addr1 == 5'b00000) && !(r_Addr2 == 5'b00000))
                nextstate = 5'b10110;
        else if ( !(r_Addr1 == 5'b00000))
                nextstate = 5'b10010;
        else
                nextstate = 5'b10001;
end

5'b10010: // Wait Low 1 state
if (remReg1[7] > 8'b00000000)
        nextstate = 5'b10100;
else if (remReg1 == 8'b00000000)
        nextstate = 5'b10011;
else
        nextstate = 5'b10010;
```

```
5'b10011: // = Low 1 state
begin
        colr = colr1;
        dot_m = read1[4:0];
        oe = 1'b1;
        cntNext = cnt + 1;
        if (cnt < 28'b1000111100001101000101111111)
                nextstate = 5'b10011;
        else if (!(cnt < 28'b1000111100001101000101111111))
                nextstate = 5'b10110;
        else
                nextstate = 5'b10011;
end

5'b10100: // > Low 1 state
begin
        colr = colr1;
        dot_m = read1[4:0];
        oe = 1'b1;
        r_Addr1Next = r_Addr1 - 1;
        ascRegNext = remReg1;
        nextstate = 5'b10101;
end

5'b10101:
begin
        cntNext = cnt + 1;
        if (cnt < 28'b1000111100001101000101111111)
                nextstate = 5'b10101;
        else if ( !(cnt < 28'b1000111100001101000101111111) &&
                r_Addr1 != 8'b00000000)
                nextstate = 5'b10001;
        else if ( cnt < 28'b1000111100001101000101111111 &&
                r_Addr1 == 8'b00000000)
                nextstate = 5'b10110;
end

5'b10110: // Wait Low 2 state
begin
        cntNext = 28'b0;
        oe = 1'b0;
        if (remReg2 == 8'b00000000)
                nextstate = 5'b11000;
        else if (remReg2 > 8'b00000000)
                nextstate = 5'b10111;
        else
                nextstate = 5'b10110;
end

5'b10111: // = Low 2 state
begin
        colr = colr2;
        dot_m = read2[4:0];
        oe = 1'b1;
        r_Addr2Next = r_Addr2 - 1;
        dscRegNext = remReg2;
        nextstate = 5'b10101;
end

5'b11000: // > Low 2 state
begin
        colr = colr2;
        dot_m = read2[4:0];
        oe = 1'b1;
        cntNext = cnt + 1;
        if (cnt < 28'b1000111100001101000101111111)
                nextstate = 5'b11000;
        else if (!(cnt < 28'b1000111100001101000101111111))
                nextstate = 5'b00000;
        else
                nextstate = 5'b11000;
end

5'b11001:
begin
        cntNext = cnt + 1;
        if (cnt < 28'b1000111100001101000101111111)
                nextstate = 5'b11001;
        else if ( !(cnt < 28'b1000111100001101000101111111) &&
                r_Addr2 == 8'b00000000)
                nextstate = 5'b00000;
        else if ( !(cnt < 28'b1000111100001101000101111111) &&
                r_Addr2 != 8'b00000000)
                nextstate = 5'b10110;
        else
                nextstate = 5'b11001;
end
```

```
                    default:
                    begin
                            ascRegNext = 8'b0;
                            dscRegNext = 8'b0;
                            cntNext = 28'b0;
                            remReg1Next = 8'b0;
                            remReg2Next = 8'b0;
                            r_Addr1Next = 5'b0;
                            r_Addr2Next = 5'b0;
                            colr = 1'b0;
                            dot_m = 5'b0;
                            oe = 1'b0;
                    end
            endcase
endmodule
```

BRAIN:M Top-Module

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Student: Mert İNAN
// Create Date:    15:31:15 12/20/2015
// Module Name:    brain_m
// Project Name:     BRAIN:M
// Target Devices:   FPGA Board
// Description: This top module uses Structural Verilog to combine different
//                    modules together to make the BRAIN:M module, which receives
//                    shape and outputs shape.
// Revision 0.01 - File Created
// Additional Comments: This top module is really huge, thus it requires a 500K
//                                       gate FPGA Board. Simulation is also an option.
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
module brain_m(clk, dist, mode, start, E, color, OE, shape);
        input clk, mode, start, E;
        input [6:0] dist;
        output color, OE;
        output [4:0] shape;
        wire [99:0] W1;
        wire W2, W3, W8, W13, W14,
                W15, W16, W17, W18, W19,
                W20, W21, W22, W23, W24,
                W25, W26, W27, W28, W29,
                W30, W31;
        wire [6:0] W4, W5;
        wire [11:0] W6, W51, W52, W53,
                    W54, W55, W56, W57,
                     W58, W59, W60, W61,
                      W62, W63, W64, W65,
                      W66, W67, W68, W69,
                      W11, W12;
        wire [4:0] W7, W32, W33, W34, W35,
                      W36, W37, W38, W39, W40,
                      W41, W42, W43, W44, W45,
                      W46, W47, W48, W49, W50,
                      W9, W10;
        //100 Sensory Neurons
        sensory_neuron s1(clk, dist, 7'b0000010, start, W1[0]);
        sensory_neuron s2(clk, dist, 7'b0000011, start, W1[1]);
        sensory_neuron s3(clk, dist, 7'b0000100, start, W1[2]);
        sensory_neuron s4(clk, dist, 7'b0000101, start, W1[3]);
        sensory_neuron s5(clk, dist, 7'b0000110, start, W1[4]);
        sensory_neuron s6(clk, dist, 7'b0000111, start, W1[5]);
        sensory_neuron s7(clk, dist, 7'b0001000, start, W1[6]);
        sensory_neuron s8(clk, dist, 7'b0001001, start, W1[7]);
        sensory_neuron s9(clk, dist, 7'b0001010, start, W1[8]);
        sensory_neuron s10(clk, dist, 7'b0001011, start, W1[9]);
        sensory_neuron s11(clk, dist, 7'b0001100, start, W1[10]);
        sensory_neuron s12(clk, dist, 7'b0001101, start, W1[11]);
        sensory_neuron s13(clk, dist, 7'b0001110, start, W1[12]);
        sensory_neuron s14(clk, dist, 7'b0001111, start, W1[13]);
        sensory_neuron s15(clk, dist, 7'b0010000, start, W1[14]);
        sensory_neuron s16(clk, dist, 7'b0010001, start, W1[15]);
        sensory_neuron s17(clk, dist, 7'b0010010, start, W1[16]);
        sensory_neuron s18(clk, dist, 7'b0010011, start, W1[17]);
        sensory_neuron s19(clk, dist, 7'b0010100, start, W1[18]);
        sensory_neuron s20(clk, dist, 7'b0010101, start, W1[19]);
        sensory_neuron s21(clk, dist, 7'b0010110, start, W1[20]);
        sensory_neuron s22(clk, dist, 7'b0010111, start, W1[21]);
        sensory_neuron s23(clk, dist, 7'b0011000, start, W1[22]);
        sensory_neuron s24(clk, dist, 7'b0011001, start, W1[23]);
        sensory_neuron s25(clk, dist, 7'b0011010, start, W1[24]);
        sensory_neuron s26(clk, dist, 7'b0011011, start, W1[25]);
        sensory_neuron s27(clk, dist, 7'b0011100, start, W1[26]);
        sensory_neuron s28(clk, dist, 7'b0011101, start, W1[27]);
        sensory_neuron s29(clk, dist, 7'b0011110, start, W1[28]);
        sensory_neuron s30(clk, dist, 7'b0011111, start, W1[29]);
        sensory_neuron s31(clk, dist, 7'b0100000, start, W1[30]);
        sensory_neuron s32(clk, dist, 7'b0100001, start, W1[31]);
        sensory_neuron s33(clk, dist, 7'b0100010, start, W1[32]);
        sensory_neuron s34(clk, dist, 7'b0100011, start, W1[33]);
        sensory_neuron s35(clk, dist, 7'b0100100, start, W1[34]);
        sensory_neuron s36(clk, dist, 7'b0100101, start, W1[35]);
        sensory_neuron s37(clk, dist, 7'b0100110, start, W1[36]);
        sensory_neuron s38(clk, dist, 7'b0100111, start, W1[37]);
        sensory_neuron s39(clk, dist, 7'b0101000, start, W1[38]);
        sensory_neuron s40(clk, dist, 7'b0101001, start, W1[39]);
        sensory_neuron s41(clk, dist, 7'b0101010, start, W1[40]);
        sensory_neuron s42(clk, dist, 7'b0101011, start, W1[41]);
        sensory_neuron s43(clk, dist, 7'b0101100, start, W1[42]);
        sensory_neuron s44(clk, dist, 7'b0101101, start, W1[43]);
        sensory_neuron s45(clk, dist, 7'b0101110, start, W1[44]);
        sensory_neuron s46(clk, dist, 7'b0101111, start, W1[45]);
        sensory_neuron s47(clk, dist, 7'b0110000, start, W1[46]);
        sensory_neuron s48(clk, dist, 7'b0110001, start, W1[47]);
        sensory_neuron s49(clk, dist, 7'b0110010, start, W1[48]);
        sensory_neuron s50(clk, dist, 7'b0110011, start, W1[49]);
        sensory_neuron s51(clk, dist, 7'b0110100, start, W1[50]);
        sensory_neuron s52(clk, dist, 7'b0110101, start, W1[51]);
        sensory_neuron s53(clk, dist, 7'b0110110, start, W1[52]);
        sensory_neuron s54(clk, dist, 7'b0110111, start, W1[53]);
        sensory_neuron s55(clk, dist, 7'b0111000, start, W1[54]);
        sensory_neuron s56(clk, dist, 7'b0111001, start, W1[55]);
        sensory_neuron s57(clk, dist, 7'b0111010, start, W1[56]);
        sensory_neuron s58(clk, dist, 7'b0111011, start, W1[57]);
        sensory_neuron s59(clk, dist, 7'b0111100, start, W1[58]);
        sensory_neuron s60(clk, dist, 7'b0111101, start, W1[59]);
        sensory_neuron s61(clk, dist, 7'b0111110, start, W1[60]);
        sensory_neuron s62(clk, dist, 7'b0111111, start, W1[61]);
        sensory_neuron s63(clk, dist, 7'b1000000, start, W1[62]);
        sensory_neuron s64(clk, dist, 7'b1000001, start, W1[63]);
        sensory_neuron s65(clk, dist, 7'b1000010, start, W1[64]);
        sensory_neuron s66(clk, dist, 7'b1000011, start, W1[65]);
        sensory_neuron s67(clk, dist, 7'b1000100, start, W1[66]);
        sensory_neuron s68(clk, dist, 7'b1000101, start, W1[67]);
        sensory_neuron s69(clk, dist, 7'b1000110, start, W1[68]);
        sensory_neuron s70(clk, dist, 7'b1000111, start, W1[69]);
        sensory_neuron s71(clk, dist, 7'b1001000, start, W1[70]);
        sensory_neuron s72(clk, dist, 7'b1001001, start, W1[71]);
        sensory_neuron s73(clk, dist, 7'b1001010, start, W1[72]);
        sensory_neuron s74(clk, dist, 7'b1001011, start, W1[73]);
        sensory_neuron s75(clk, dist, 7'b1001100, start, W1[74]);
        sensory_neuron s76(clk, dist, 7'b1001101, start, W1[75]);
        sensory_neuron s77(clk, dist, 7'b1001110, start, W1[76]);
        sensory_neuron s78(clk, dist, 7'b1001111, start, W1[77]);
        sensory_neuron s79(clk, dist, 7'b1010000, start, W1[78]);
        sensory_neuron s80(clk, dist, 7'b1010001, start, W1[79]);
        sensory_neuron s81(clk, dist, 7'b1010010, start, W1[80]);
        sensory_neuron s82(clk, dist, 7'b1010011, start, W1[81]);
        sensory_neuron s83(clk, dist, 7'b1010100, start, W1[82]);
        sensory_neuron s84(clk, dist, 7'b1010101, start, W1[83]);
        sensory_neuron s85(clk, dist, 7'b1010110, start, W1[84]);
        sensory_neuron s86(clk, dist, 7'b1010111, start, W1[85]);
        sensory_neuron s87(clk, dist, 7'b1011000, start, W1[86]);
        sensory_neuron s88(clk, dist, 7'b1011001, start, W1[87]);
        sensory_neuron s89(clk, dist, 7'b1011010, start, W1[88]);
        sensory_neuron s90(clk, dist, 7'b1011011, start, W1[89]);
        sensory_neuron s91(clk, dist, 7'b1011100, start, W1[90]);
        sensory_neuron s92(clk, dist, 7'b1011101, start, W1[91]);
        sensory_neuron s93(clk, dist, 7'b1011110, start, W1[92]);
        sensory_neuron s94(clk, dist, 7'b1011111, start, W1[93]);
        sensory_neuron s95(clk, dist, 7'b1100000, start, W1[94]);
        sensory_neuron s96(clk, dist, 7'b1100001, start, W1[95]);
```

```
    sensory_neuron s97(clk, dist, 7'b1100010, start, W1[96]);
    sensory_neuron s98(clk, dist, 7'b1100011, start, W1[97]);
    sensory_neuron s99(clk, dist, 7'b1100100, start, W1[98]);
    sensory_neuron s100(clk, dist, 7'b1100101, start, W1[99]);

    //2 Direction Ganglions
    direction_ganglion d1(clk, W1, start, 1'b1, W2, W4);
    direction_ganglion d2(clk, W1, start, 1'b0, W3, W5);

    //20 Interneurons
    inter_neuron in1(clk, W5, 7'b1100100, mode, W6, W7, W8);
    inter_neuron in2(clk, W5, 7'b1011111, mode, W51, W32, W13);
    inter_neuron in3(clk, W5, 7'b1011010, mode, W52, W33, W14);
    inter_neuron in4(clk, W5, 7'b1010101, mode, W53, W34, W15);
    inter_neuron in5(clk, W5, 7'b1010000, mode, W54, W35, W16);
    inter_neuron in6(clk, W5, 7'b1001011, mode, W55, W36, W17);
    inter_neuron in7(clk, W5, 7'b1000110, mode, W56, W37, W18);
    inter_neuron in8(clk, W5, 7'b1000001, mode, W57, W38, W19);
    inter_neuron in9(clk, W5, 7'b0111100, mode, W58, W39, W20);
    inter_neuron in10(clk, W5, 7'b0110111, mode, W59, W40, W21);
    inter_neuron in11(clk, W5, 7'b0110010, mode, W60, W41, W22);
    inter_neuron in12(clk, W5, 7'b0101101, mode, W61, W42, W23);
    inter_neuron in13(clk, W5, 7'b0101000, mode, W62, W43, W24);
    inter_neuron in14(clk, W5, 7'b0100011, mode, W63, W44, W25);
    inter_neuron in15(clk, W5, 7'b0011110, mode, W64, W45, W26);
    inter_neuron in16(clk, W5, 7'b0011001, mode, W65, W46, W27);
    inter_neuron in17(clk, W5, 7'b0010100, mode, W66, W47, W28);
    inter_neuron in18(clk, W5, 7'b0001111, mode, W67, W48, W29);
    inter_neuron in19(clk, W5, 7'b0001010, mode, W68, W49, W30);
    inter_neuron in20(clk, W5, 7'b0000101, mode, W69, W50, W31);

    //Memory Neuron Array
    memory_neuron m(clk, W8, W13, W14, W15, W16, W17, W18, W19,
                    W20, W21, W22, W23, W24, W25, W26, W27, W28,
                    W29, W30, W31, W7, W32, W33, W34, W35, W36,
                    W37, W38, W39, W40, W41, W42, W43, W44, W45,
                    W46, W47, W48, W49, W50, W6, W51, W52, W53,
                    W54, W55, W56, W57, W58, W59, W60, W61, W62,
                    W63, W64, W65, W66, W67, W68, W69, W9, W10,
                    W11, W12);
    //Intelligent Stimuli-Comparison Unit
    intelligent_stimuli_comparison_unit iscu(clk, E, W4, W5, mode, W11, W12,
                                             W2, W3, color, OE, shape, W9, W10);

endmodule
```

Sample Test Bench

```
////////////////////////////////////////////////////////////////////////////////
// Student: Mert İnan
//
// Create Date:   21:28:22 12/21/2015
// Design Name:   direction_ganglion
// Module Name:   /home/merterm/Desktop/brain_M/direction_ganglion_test.v
// Project Name:  brain_M
// Description:
//
// Verilog Test Fixture created by ISE for module: direction_ganglion
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module direction_ganglion_test;

	// Inputs
	reg clk;
	reg [99:0] dist;
	reg start;
	reg dir;

	// Outputs
	wire colrcode;
	wire [6:0] flat;

	// Instantiate the Unit Under Test (UUT)
	direction_ganglion uut (
		.clk(clk),
		.dist(dist),
		.start(start),
		.dir(dir),
		.colrcode(colrcode),
		.flat(flat)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
```

```
                dist = 0;
                start = 0;
                dir = 0;

                // Wait 100 ns for global reset to finish
                #100;

                // Add stimulus here
                dir = 1;
                start = 1;
                dist = 1;
                #55000000
                dist = 2;
                #55000000
                dist = 4;
                //#100000000
                //dist = 8;
                //#100000000
                //dist = 16;
                //#100000000
                //dist = 32;
                //#100000000
                //dist = 64;
                //#100000000
                //dist = 128;
                //#100000000
                //dist = 256;
                //#100000000
                //dist = 512;
                #55000000
                start = 0;
                #55000000
                dist = 0;

        end
   always
                #1 clk = !clk;
endmodule
```

```
NET "CLK" LOC = "B8";
NET "color" LOC = "N3";
NET "shape[0]" LOC = "E2";
NET "shape[1]" LOC = "F3";
NET "shape[2]" LOC = "G3";
NET "shape[3]" LOC = "B4";
NET "shape[4]" LOC = "K3";

NET"DS"LOC="A9";
NET"OE"LOC="B9";
NET"STCP"LOC="A10";
NET"SHCP"LOC="C9";
NET"MR"LOC="C12";

NET"K[7]"LOC="B2";
NET"K[6]"LOC="A3";
NET"K[5]"LOC="J3";
NET"K[4]"LOC="B5";
NET"K[3]"LOC="C6";
NET"K[2]"LOC="B6";
NET"K[1]"LOC="C5";
NET"K[0]"LOC="B7";
NET"d"LOC="G1";

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Student: İmge Gökalp
// Create Date:    18:49:33 12/20/2015
// Module Name:    dot_matrix_final
// Project Name:     BRAIN:M
// Target Devices:   FPGA Board
// Description:       This module creates 20 different shapes on the dot matrix
//                          according to its inputs.
// Revision 0.01 - File Created
//////////////////////////////////////////////////////////////////////////////////
module dot_matrix_final( color, shape, CLK, OE, SHCP, STCP, MR, DS, K, d);

	input CLK, color;
	input [4:0] shape;
	output reg OE;
	output reg SHCP, STCP, MR, DS;
	output reg [7:0] K;
	reg [23:0] signal;
	reg [7:0] R, B, G, counter, a;
	reg f, e;
reg [8:0] i;
output reg d;


always @ (posedge CLK)
    begin
        d<=1;
        counter <= counter + 1;
    end

always @(posedge e)
    begin
        if (i < 9'b110100011)
            i = i + 1;
        else
            i = 9'b000000000;
    end

always@(*)
    begin

        signal[23:16] <= R;
        signal[15:8] <= G;
        signal[7:0] <= B;

        f <= counter[7];
        e <= ~f;

        if (i < 9'b000000011)
            MR = 0;
        else
            MR = 1;

        if (i > 9'b000000010 && i < 9'b000011011)
            DS = signal[i-3];
        else
            DS = 0;

        if (i < 9'b000011011)
            begin
                SHCP = f;
                STCP = e;
            end
        else
            begin
```

```
                SHCP = 0;
                STCP = 1;
            end

        if (a == 8'b00000000)
            K = 8'b10000000;
        else if (a == 8'b00000001)
            K = 8'b01000000;
        else if (a == 8'b00000010)
            K = 8'b00100000;
        else if (a == 8'b00000011)
            K = 8'b00010000;
        else if (a == 8'b00000100)
            K = 8'b00001000;
        else if (a == 8'b00000101)
            K = 8'b00000100;
        else if (a == 8'b00000110)
            K = 8'b00000010;
        else
            K = 8'b00000001;

    end

always @ (posedge f)
    begin
        if (i > 9'b000011011 && i < 9'b110011000)
            OE = 0;
        else
            OE= 1;

        if (i == 9'b110011001)
            begin
                if (a < 8'b00001000)
                    a = a + 1;
                else
                    a = 8'b00000000;
            end
        else
            a = a;
    end

always @ (*)
    case(shape)
        5'b00000:
            begin
```

```
if(color)
    begin
        if (a == 0)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 1)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b01000010;
            end
        else if (a == 2)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b00100100;
            end
        else if (a == 3)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b00011000;
            end
        else if (a == 4)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b00011000;
            end
        else if (a == 5)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b00100100;
            end
        else if (a == 6)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b01000010;
            end
        else
```

```
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
    end
else
    begin
        if (a == 0)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 1)
            begin
                R <= 8'b01000010;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 2)
            begin
                R <= 8'b00100100;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 3)
            begin
                R <= 8'b00011000;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 4)
            begin
                R <= 8'b00011000;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 5)
            begin
                R <= 8'b00100100;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 6)
```

```
                    begin
                        R <= 8'b01000010;
                        B <= 8'b00000000;
                        G <= 8'b00000000;
                    end
                else
                    begin
                        R <= 8'b00000000;
                        B <= 8'b00000000;
                        G <= 8'b00000000;
                    end
            end
        end
5'b00001:
    begin
        if(color)
            begin
                if (a == 0)
                    begin
                        R <= 8'b00000000;
                        B <= 8'b00000000;
                        G <= 8'b11111111;
                    end
                else if (a == 1)
                    begin
                        R <= 8'b00000000;
                        B <= 8'b00000000;
                        G <= 8'b10000001;
                    end
                else if (a == 2)
                    begin
                        R <= 8'b00000000;
                        B <= 8'b00000000;
                        G <= 8'b10000001;
                    end
                else if (a == 3)
                    begin
                        R <= 8'b00000000;
                        B <= 8'b00000000;
                        G <= 8'b10000001;
                    end
                else if (a == 4)
                    begin
                        R <= 8'b00000000;
                        B <= 8'b00000000;
                G <= 8'b10000001;
            end
        else if (a == 5)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b10000001;
            end
        else if (a == 6)
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b10000001;
            end
        else
            begin
                R <= 8'b00000000;
                B <= 8'b00000000;
                G <= 8'b11111111;
            end
    end
else
    begin
        if (a == 0)
            begin
                R <= 8'b11111111;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 1)
            begin
                R <= 8'b10000001;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 2)
            begin
                R <= 8'b10000001;
                B <= 8'b00000000;
                G <= 8'b00000000;
            end
        else if (a == 3)
            begin
                R <= 8'b10000001;
                B <= 8'b00000000;
```

```
                                    G <= 8'b00000000;
                                end
                            else if (a == 4)
                                begin
                                    R <= 8'b10000001;
                                    B <= 8'b00000000;
                                    G <= 8'b00000000;
                                end
                            else if (a == 5)
                                begin
                                    R <= 8'b10000001;
                                    B <= 8'b00000000;
                                    G <= 8'b00000000;
                                end
                            else if (a == 6)
                                begin
                                    R <= 8'b10000001;
                                    B <= 8'b00000000;
                                    G <= 8'b00000000;
                                end
                            else
                                begin
                                    R <= 8'b11111111;
                                    B <= 8'b00000000;
                                    G <= 8'b00000000;
                                end
                        end
                end
            5'b00010:
                begin
                    if(color)
                        begin
                            if (a == 0)
                                begin
                                    R <= 8'b00000000;
                                    B <= 8'b00000000;
                                    G <= 8'b11111111;
                                end
                            else if (a == 1)
                                begin
                                    R <= 8'b00000000;
                                    B <= 8'b00000000;
                                    G <= 8'b11111111;
                                end
                            else if (a == 2)
                begin
                    R <= 8'b00000000;
                    B <= 8'b00000000;
                    G <= 8'b11111111;
                end
            else if (a == 3)
                begin
                    R <= 8'b00000000;
                    B <= 8'b00000000;
                    G <= 8'b11111111;
                end
            else if (a == 4)
                begin
                    R <= 8'b00000000;
                    B <= 8'b00000000;
                    G <= 8'b11111111;
                end
            else if (a == 5)
                begin
                    R <= 8'b00000000;
                    B <= 8'b00000000;
                    G <= 8'b11111111;
                end
            else if (a == 6)
                begin
                    R <= 8'b00000000;
                    B <= 8'b00000000;
                    G <= 8'b11111111;
                end
            else
                begin
                    R <= 8'b00000000;
                    B <= 8'b00000000;
                    G <= 8'b11111111;
                end
        end
    else
        begin
            if (a == 0)
                begin
                    R <= 8'b11111111;
                    B <= 8'b00000000;
                    G <= 8'b00000000;
                end
        else if (a == 1)
```

```
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b00011:
begin
```

```
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else
begin
```

```
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b01111110;
```

```
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b00100:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 5)
```

```
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
```

```
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b00101:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
```

```
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
```

```
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b00110:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
```

```
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
```

```
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
```

```
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b00111:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
```

```
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
```

```
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01000:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
```

```
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
```

```
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01001:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
```

```
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
```

```
 else if (a == 1)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01010:
```

```
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
```

```
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
```

```
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01011:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
```

```
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
```

```
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01100:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
```

```
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
```

```
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01101:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
```

```
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
```

```
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
```

```
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01110:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
```

```
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
```

```
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b01111:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
```

```
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
```

```
 else if (a == 3)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b10000:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
```

```
G <= 8'b01000010;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00111100;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
```

```
end
 else if (a == 1)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00111100;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
```

```
5'b10001:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b11111111;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
```

```
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b11111111;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
```

```
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b10010:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01111110;
```

```
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
```

```
 else if (a == 4)
begin
R <= 8'b01111110;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b10011:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
```

```
G <= 8'b00100100;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b10000001;
end
end
else begin
 if (a == 0)
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
```

```
end
 else if (a == 2)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
begin
R <= 8'b10000001;
B <= 8'b00000000;
G <= 8'b00000000;
end
end
end
5'b10100:
begin
if(color) begin
 if (a == 0)
begin
R <= 8'b00000000;
```

```
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else if (a == 2)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 3)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 4)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00011000;
end
 else if (a == 5)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00100100;
end
 else if (a == 6)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b01000010;
end
 else
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
```

```
end
else begin
 if (a == 0)
begin
R <= 8'b00000000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 1)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 2)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 3)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 4)
begin
R <= 8'b00011000;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 5)
begin
R <= 8'b00100100;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else if (a == 6)
begin
R <= 8'b01000010;
B <= 8'b00000000;
G <= 8'b00000000;
end
 else
```

```
                                    begin
                                    R <= 8'b00000000;
                                    B <= 8'b00000000;
                                    G <= 8'b00000000;
                                    end
                                end
                            end
        default:
            begin
            R <= 8'b00000000;
            B <= 8'b00000000;
            G <= 8'b00000000;
            end

    endcase
endmodule
```

Ultrasound PING))) Module

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Student: İmge Gökalp
// Create Date:    17:46:20 12/22/2015
// Module Name:    ultrasonic_measurement
// Project Name:     BRAIN:M
// Target Devices:   FPGA Board
// Description:  This is the ultrasound controller module
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
module ultrasonic_measurement( rst, CLK, ptime, sig);

    input rst, clk;
    output reg [19:0] ptime;
    output reg sig;

    reg [25:0] counter;
    reg [19:0] distcalc, value;
    reg [1:0] state, staterd;
    reg sigbuf, sigbuf2, sigbuf3;

    ptime <= value;

    always@(rst or CLK)
        begin
            if(~rst)
                begin
                    ptime = 20'b0;
                    sig = 0;
                    counter = 26'b0;
                    distcalc = 20'b0;
                    value = 20'b0;
                    state = 2'b0;
                    staterd = 2'b0;
                    sigbuf = 0;
                    sigbuf2 = 0;
                    sigbuf3 = 0;
                end
            else
                sig = sig;
        end
```

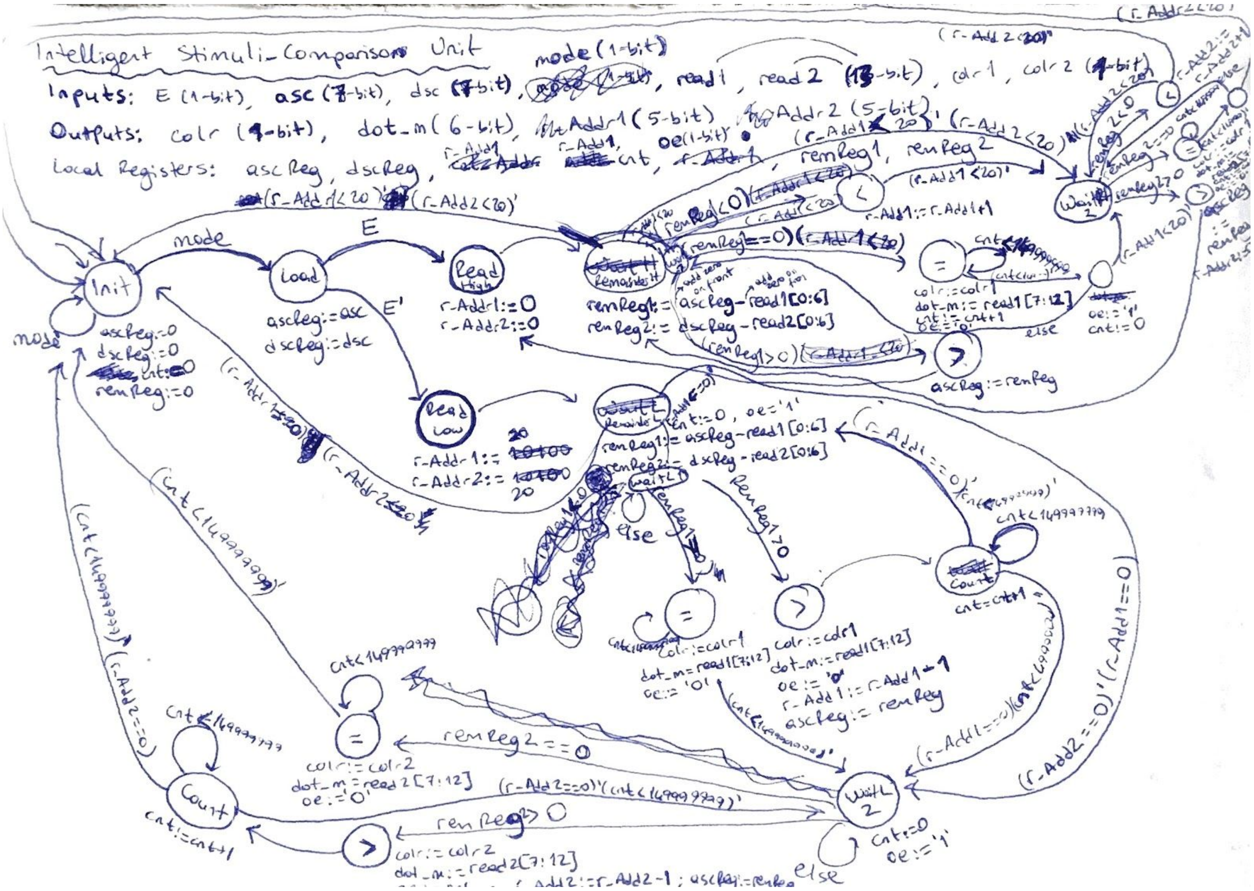
```
always@(posedge CLK)
    begin
        counter = counter + 1;

        case(state)
            2'00:
                begin
                    distcalc = 20'b0;
                    staterd = 2'b00;
                    sig = 0;

                    if(counter == 50000000)
                        begin
                            state = 2'b01;
                            counter = 0;
                        end
                end
            2'01:
                begin
                    distcalc = 20’b0;
                    sig = 1;

                    if(counter == 1000)
                        begin
                            state = 11;
                            counter = 0;
                        end

                end
            2’b11:
                begin
                    distcalc = 20’b0;
                    sig = 0;
                    staterd = 2’b00;

                    if(counter ==           50000)
                        begin
                            state = 2’b10;
                            counter = 0;
                        end
                end
            2’b10:
                begin
                    sig = Z;
```

```
sigbuf = sig;
sigbuf2 = sigbuf;
sigbuf3 = sigbuf2;

case(staterd)
    2’b00:begin
        distcalc = 20’b0;

        if(sigbuf3 == 0 && sigbuf2 ==1)
            begin
                staterd = 2’b01;
                counter = 0;
            end

        if(counter == 100000)
            begin
                staterd = 2’b11;
                counter = 0;
            end end

    2’b01: begin
        distcalc = distcalc + 1;

        if(sigbuf3 == 1 && sigbuf2 ==0)
            begin
                staterd = 2’b11;
                counter = 0;
            end

        if(counter == 100000)
            begin
                staterd = 2’b11;
                distcalc = 20’b0;
                counter = 0;
            end end

    2’b11: begin
        value = distcalc[19:0];
        counter = 0;
        state = 2’b00;
        staterd = 2’b00;

    default:
        staterd = 2’b00;
```

```
                    end
                default:
                    state = 2’b00;
        end


endmodule
```

BONUS

Intelligent Stimuli Comparison Unit HLSM Draft


Intelligent Stimuli-Comparison Unit
Inputs: E (1-bit), asc (7-bit), dsc (7-bit), mode (1-bit), read1, read2 (13-bit), colr1, colr2 (9-bit)
Outputs: colr (9-bit), dot_m (6-bit), r_Addr1 (5-bit), r_Addr2 (5-bit), oe (1-bit)
Local Registers: ascReg, dscReg, r_Add1, r_Add1, cnt, remReg1, remReg2
(r_Add1<20)' (r_Add2<20)'
mode
E
E'
Init
Load
Read High
Read Low
mode
ascReg:=0
dscReg:=0
cnt:=0
remReg:=0
ascReg:=asc
dscReg:=dsc
r_Addr1:=0
r_Addr2:=0
r_Addr1:= 20
r_Addr2:= 20
Remainder H
remReg1:= ascReg-read1[0:6]
remReg2:= dscReg-read2[0:6]
add zero on front
add zero on front
remReg<0
(r_Add1<20)
remReg==0
(remReg1>0)
<
r_Add1:=r_Add1+1
(r_Add1<20)'
=
colr:=colr1
dot_m:=read1[7:12]
cnt:=cnt+1
oe:='0'
cnt<149999999
else
oe:='1'
cnt:=0
>
ascReg:=remReg
(r_Add1<20)'
Wait 2
(r_Add2<20)'
(r_Add2<20)
remReg2<0
remReg2==0
remReg2>0
r_Add2:=r_Add2+1
else
Remainder L
cnt:=0, oe='1'
remReg1:= ascReg-read1[0:6]
remReg2:= dscReg-read2[0:6]
(r_Add1==0)'
(r_Addr1==0)
(r_Addr2==0)
wait L1
else
remReg1==0
remReg1>0
=
colr:=colr1
dot_m=read1[7:12]
oe:='0'
>
colr:=colr1
dot_m:=read1[7:12]
oe:='0'
r_Add1:=r_Add1-1
ascReg:=remReg
Count
cnt<149999999
cnt=cnt+1
(cnt<149999999)'
(r_Add1==0)(cnt<149999999)
(r_Add2==0)'(r_Add1==0)
(cnt<149999999)'
(cnt<149999999)'
(r_Add2==0)
cnt<149999999
=
colr:=colr2
dot_m=read2[7:12]
oe:='0'
remReg2==0
(r_Add2==0)'(cnt<149999999)'
Count
cnt<149999999
cnt:=cnt+1
remReg2>0
>
colr:=colr2
dot_m:=read2[7:12]
r_Add2:=r_Add2-1 ; ascReg:=remReg
Wait L 2
cnt:=0
oe:='1'
else