Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory

Gary M. Matthias, Timothy M. Mwangi and Anthony J. Quivers

May 17, 2007

Design and Implementation of a 3-D game of pool on an FPGA Using the Major-Minor FSM Setup

Appendix

```
// Gary Matthias
// Decodes NTSC video by waiting for a specific field change
module pixel_receiver(reset, clock, tv_in_ycrcb, read_pixel_count,
                      read_line_count, tv_in_chroma_blue,
                      tv_in_chroma_red, tv_in_luma, position_known,
                      pixel_ready, debug);


   // Parameters
   parameter STATE_CHROMA_BLUE = 2'd3;
   parameter STATE_LUMA_1 = 2'd2;
   parameter STATE_CHROMA_RED = 2'd1;
   parameter STATE_LUMA_2 = 2'd0;

   // Inputs
   input reset, clock;
   input[7:0] tv_in_ycrcb;

   // Input Wires
   wire reset, clock;
   wire[7:0] tv_in_ycrcb;

   // Outputs
   output[9:0] read_pixel_count, read_line_count;
   output[7:0] tv_in_chroma_blue, tv_in_luma,
               tv_in_chroma_red;
   output position_known, pixel_ready;
   output[31:0] debug;

   // Output Registers
   reg[9:0] read_pixel_count, read_line_count;
   reg[7:0] tv_in_chroma_blue, tv_in_luma,
            tv_in_chroma_red;
   reg position_known, pixel_ready;
        wire[31:0] debug;

   // Parameters
   parameter REFERENCE_CODE = 8'hFF;
   parameter PIXEL_MAX = 10'd857;
   parameter LINE_MAX = 10'd524;

   // Other registers
   // Set to 1 if chroma was all ones and
   // luma was all zeroes on previous clock cycle
   reg f;
   reg v;
   reg h;
   reg[1:0] count;
   reg[1:0] currentstate;


   assign debug = {3'b000,1'b0,
                   3'b000,f,
                   3'b000,v,
                   3'b000,h,
                   2'b00,count,
                   2'b00,currentstate,
                   4'b0000};

   always @(posedge clock) begin

      if (tv_in_ycrcb == 8'hFF) begin
         count <= 2'd3;
      end
      else if (count > 2'd1) begin
         count <= count - 1;
      end
      else if (count == 2'd1) begin
         f <= tv_in_ycrcb[6];
         v <= tv_in_ycrcb[5];
         h <= tv_in_ycrcb[4];
         count <= 2'b0;
      end
      else begin end

      if (reset) begin
         read_pixel_count <= 10'd0;
         read_line_count <= 10'd0;
         position_known <= 1'b0;
         count <= 2'b0;
         currentstate <= 2'd0;
         tv_in_chroma_blue <= 8'b0;
         tv_in_luma <= 8'b0;
         tv_in_chroma_red <= 8'b0;
         f <= 1'b0;
         v <= 1'b0;
         h <= 1'b0;
         pixel_ready <= 1'b0;
      end
      else if ((count == 2'd1) & ~position_known) begin
         // tv_in_luma is in XY mode if a reference code was received on the
         // last cycle
         // It has format: 1, f, v, h, p3, p2, p1, p0
         //    p3 = v ^ h
         //    p2 = f ^ h
         //    p1 = f ^ v
         //    p0 = f ^ v ^ h
         if (({f,v,h} == 3'b010) & (tv_in_ycrcb[6:4] == 3'b001)) begin
```

```
            read_pixel_count <= 10'd721;
            read_line_count <= 10'd18;
            position_known <= 1'b1;
            currentstate <= STATE_CHROMA_BLUE;
         end
         else begin end
      end
      // If position not found yet, wait for next clock cycle
      else if (~position_known) begin end
      // Store chroma blue
      else if (currentstate == STATE_CHROMA_BLUE) begin
         tv_in_chroma_blue <= tv_in_ycrcb;
         currentstate <= STATE_LUMA_1;
         read_pixel_count <= (read_pixel_count==PIXEL_MAX) ?
            10'd0 : read_pixel_count+1;
         read_line_count <= (read_pixel_count==PIXEL_MAX) ?
             ( (read_line_count==LINE_MAX) ? 10'd0 : read_line_count+1) :
             read_line_count;
         pixel_ready <= 1'b0;
      end
      // Store luma
      else if (currentstate == STATE_LUMA_1) begin
         tv_in_luma <= tv_in_ycrcb;
         currentstate <= STATE_CHROMA_RED;
         pixel_ready <= 1'b1;
      end
      // Store chroma red
      else if (currentstate == STATE_CHROMA_RED) begin
         tv_in_chroma_red <= tv_in_ycrcb;
         currentstate <= STATE_LUMA_2;
         read_pixel_count <= (read_pixel_count==PIXEL_MAX) ?
            10'd0 : read_pixel_count+1;
         read_line_count <= (read_pixel_count==PIXEL_MAX) ?
             ( (read_line_count==LINE_MAX) ? 10'd0 : read_line_count+1) :
             read_line_count;
         pixel_ready <= 1'b0;
      end
      // Store luma
      else if (currentstate == STATE_LUMA_2) begin
         tv_in_luma <= tv_in_ycrcb;
         currentstate <= STATE_CHROMA_BLUE;
         pixel_ready <= 1'b1;
      end
      else begin end
   end

endmodule

// Gary Matthias
// Transforms the NTSC line and pixel to the line and pixel it represents
// if the line and pixel are active
module real_location(reset, clock, position_known, pixel_ready_in,
   tv_in_chroma_blue, tv_in_chroma_red, tv_in_luma, read_pixel_count,
      read_line_count, pixel_count, line_count, inactive_pixel, chroma_blue_reg,
      chroma_red_reg, luma_reg, location_ready_out, debug);

   // Inputs
   input reset, clock, position_known, pixel_ready_in;
   input[9:0] read_pixel_count, read_line_count;
```

```
input[7:0] tv_in_chroma_blue, tv_in_chroma_red, tv_in_luma;

// Wire Inputs
wire reset, clock, position_known, pixel_ready_in;
wire[9:0] read_pixel_count, read_line_count;
wire[7:0] tv_in_chroma_blue, tv_in_chroma_red, tv_in_luma;

// Outputs
output[9:0] pixel_count, line_count, inactive_pixel;
output[7:0] chroma_blue_reg, chroma_red_reg, luma_reg;
   output[31:0] debug;
   output location_ready_out;

// Output Registers
reg[9:0] pixel_count, line_count;
reg[7:0] chroma_blue_reg, chroma_red_reg, luma_reg;
   reg location_ready_out;

// Output Wires
parameter[9:0] INACTIVE_PIXEL_VALUE = 10'h3ff;
wire[9:0] inactive_pixel = INACTIVE_PIXEL_VALUE;

// Pixel Logic
   // On active lines, the active pixels are 0 through 719, but
   //    pixel 0 is ignored because it has incomplete data
   // The active lines are from 19 to 261 (even lines starting from 0)
   //    and from 282 to 524 (odd lines starting from 1)
wire pixel_active = (read_pixel_count>=10'd1) & (read_pixel_count<10'd720);
wire line_active_1 = (read_line_count>=10'd19) & (read_line_count<10'd262);
wire line_active_2 = (read_line_count>=10'd282) & (read_line_count<10'd524);
wire output_active = position_known & pixel_active &
                     (line_active_1 | line_active_2);

   // Debugging Assignments
   wire[31:0] debug = {16'b0,
                   3'b0,pixel_active,
                                  3'b0,line_active_1,
                               3'b0,line_active_2,
                3'b0,output_active};

always @(posedge clock) begin
   if (reset) begin
      pixel_count <= INACTIVE_PIXEL_VALUE;
      line_count <= INACTIVE_PIXEL_VALUE;
               chroma_blue_reg <= 1'b0;
               chroma_red_reg <= 1'b0;
               luma_reg <= 1'b0;
               location_ready_out <= 1'b0;
   end
   else if (pixel_ready_in & output_active) begin
      pixel_count <= read_pixel_count;
      line_count <= line_active_1 ?
                    ((read_line_count-19)*2) :
                    (((read_line_count-282)*2)+1);
               chroma_blue_reg <= tv_in_chroma_blue;
               chroma_red_reg <= tv_in_chroma_red;
         luma_reg <= tv_in_luma;
               location_ready_out <= 1'b1;
   end
```

```
      else begin
         pixel_count <= INACTIVE_PIXEL_VALUE;
         line_count <= INACTIVE_PIXEL_VALUE;
                  location_ready_out <= 1'b0;
      end
   end

endmodule


// Gary Matthias
// Determines whether a pixel meets specific thresholds for chroma blue,
// chroma red, and luma
module color_threshold(reset, clock, enable, luma_min, luma_max,
                       chroma_red_min, chroma_red_max,
                       chroma_blue_min, chroma_blue_max, pixel_count_in,
                       line_count_in, inactive_pixel, luma_in, chroma_red_in,
                       chroma_blue_in, pixel_count_out, line_count_out,
                       meets_threshold, start_cycle);

   // Inputs
   input reset, clock, enable;
   input[7:0] luma_min, luma_max, luma_in,
              chroma_red_min, chroma_red_max, chroma_red_in,
              chroma_blue_min, chroma_blue_max, chroma_blue_in;
   input[9:0] pixel_count_in, line_count_in, inactive_pixel;

   // Input Wires
   wire reset, clock, enable;
   wire[7:0] luma_min, luma_max, luma_in,
             chroma_red_min, chroma_red_max, chroma_red_in,
             chroma_blue_min, chroma_blue_max, chroma_blue_in;
   wire[9:0] pixel_count_in, line_count_in, inactive_pixel;

   // Outputs
   output meets_threshold, start_cycle;
   output[9:0] pixel_count_out, line_count_out;

   // Output Registers
   reg meets_threshold, start_cycle;
   reg[9:0] pixel_count_out, line_count_out;

   // Assignments
   wire chroma_logic = (chroma_blue_in >= chroma_blue_min) &
                       (chroma_blue_in < chroma_blue_max) &
                       (chroma_red_in >= chroma_red_min) &
                       (chroma_red_in < chroma_red_max);
   wire luma_logic = (luma_in >= luma_min) &
                     (luma_in < luma_max);
   wire valid_pixel = (pixel_count_in != inactive_pixel) &
                      (line_count_in != inactive_pixel);


   always @(posedge clock) begin

      if (reset) begin
         pixel_count_out <= inactive_pixel;
         line_count_out <= inactive_pixel;
         meets_threshold <= 1'b0;
```

```
                  start_cycle <= 1'b0;
      end
      else if (enable) begin
         pixel_count_out <= pixel_count_in;
         line_count_out <= line_count_in;
         meets_threshold <= valid_pixel & luma_logic & chroma_logic;

         if ((pixel_count_in == 10'd719) & (line_count_in == 10'd483)) begin
            start_cycle <= 1'b1;
         end
         else begin
            start_cycle <= 1'b0;
         end
      end
      else begin
         pixel_count_out <= inactive_pixel;
         line_count_out <= inactive_pixel;
         meets_threshold <= 1'b0;
         start_cycle <= 1'b0;
      end
   end

endmodule

module point_summation(reset, clock, sum_reset, enable_in, weight_in,
                       total_weight, number_of_points, enable_out, debug);

   input reset, clock, sum_reset, enable_in;
   input[9:0] weight_in;

      wire reset, clock, sum_reset, enable_in;
   wire[9:0] weight_in;

   output[26:0] total_weight;
   output[18:0] number_of_points;
      output enable_out;

   reg[26:0] total_weight, accumulated_weight;
   reg[18:0] number_of_points, accumulated_points;
      reg enable_out;

   output[31:0] debug;
      wire[31:0] debug = accumulated_points;

   always @(posedge clock) begin
      if (reset) begin
         accumulated_weight <= 27'b0;
         accumulated_points <= 19'b0;
         total_weight <= 27'b0;
         number_of_points <= 19'b0;
                  enable_out <= 1'b0;
      end
      else if (sum_reset) begin
         total_weight <= enable_in ?
                                    accumulated_weight + weight_in :
                                                       accumulated_weight;
         number_of_points <= enable_in ?
                                        accumulated_points + 1'b1 :
                                        accumulated_points;
         accumulated_weight <= 27'b0;
         accumulated_points <= 19'b0;
                  enable_out <= 1'b1;
      end
      else if (enable_in) begin
         accumulated_weight <= accumulated_weight + weight_in;
         accumulated_points <= accumulated_points + 1'b1;
                  enable_out <= 1'b0;
      end
      else begin
         enable_out <= 1'b0;
      end
   end

endmodule

// Gary Matthias
// Divides a 27-bit number by a 19-bit number
module division(reset, clock, start, dividend, divisor,
                ready, quotient, remainder, debug);

   // Inputs
   input reset, clock, start;
   input [26:0] dividend;
   input [18:0] divisor;

   // Input Wires
   wire reset, clock, start;
   wire [26:0] dividend;
   wire [18:0] divisor;

   // Outputs
   output ready;
   output [26:0] quotient, remainder;
      output [45:0] debug;

   // Output Registers
   reg ready;
   reg [26:0] quotient, remainder;
      //reg [45:0] debug;

   // Other Registers
   reg[4:0] bit;
   //reg [45:0] difference;
   reg [45:0] dividend_copy, divisor_copy;
   reg [26:0] quotient_temp;

   wire [45:0] debug = dividend_copy;
   wire signed [45:0] difference = dividend_copy - divisor_copy;

   always @(posedge clock) begin
      if (reset) begin
         quotient <= 27'b0;
         remainder <= 27'b0;
         bit <= 5'b0;
         dividend_copy <= {19'b0, dividend};
         divisor_copy <= {1'b0, divisor, 26'b0};
                  ready <= 1'b1;
      end
      else if (ready & start) begin
               dividend_copy <= {19'b0, dividend};
         divisor_copy <= {1'b0, divisor, 26'b0};
         bit <= 5'd27;
                  ready <= 1'b0;
      end
            else if (bit == 5'b0) begin
                  quotient <= quotient_temp;
                  remainder <= dividend_copy[26:0];
                  ready <= 1'b1;
            end
      else begin
                  //difference <= dividend_copy - divisor_copy;
                  quotient_temp <= quotient_temp << 1;

                  if (bit == 5'b0) begin
                        //quotient <= quotient_temp << 1;
                        //remainder <= dividend_copy[26:0];
                        //ready <= 1'b1;
                  end

                  if ( ~difference[45] ) begin
                        dividend_copy <= difference;
                        quotient_temp[0] <= 1'b1;

                        if (bit == 5'b0) begin
                              //quotient[0] <= 1'b1;
                              //remainder <= difference[26:0];
                        end
                  end

                  divisor_copy <= divisor_copy >> 1;
                  bit <= bit - 5'b1;

            end
   end


endmodule

// Gary Matthias
// Stores the input from the analog-to-digital converter (AD7810Y)
module adc_input(reset, clock_in, adc_in, start, clock_out, acc_out);

   input reset, clock_in, adc_in;
      wire reset, clock_in, adc_in;

      output start;
      reg start;

      output[9:0] acc_out;
      reg[9:0] acc_out;

      output clock_out;
      reg clock_out;

      reg[9:0] acc_out_temp;
      reg[6:0] count;
```

```
always @(posedge clock_in) begin
      if (reset) begin
            start <= 1'b1;
            count <= 7'd0;
            clock_out <= 1'b0;
            acc_out_temp <= 10'b0;
      end
      else if (count == 7'd0) begin
            start <= 1'b1;
            acc_out <= acc_out_temp;
            count <= count + 1;
      end
      else if (count == 7'd75) begin
            clock_out <= 1'b1;
            count <= count + 1;
      end
      else if (count >= 7'd76 & count <= 7'd96) begin
            clock_out <= count[0];

            if (count == 7'd77) begin
               acc_out_temp[9] <= adc_in;
            end
            else if (count == 7'd79) begin
               acc_out_temp[8] <= adc_in;
            end
            else if (count == 7'd81) begin
               acc_out_temp[7] <= adc_in;
            end
            else if (count == 7'd83) begin
               acc_out_temp[6] <= adc_in;
            end
            else if (count == 7'd85) begin
               acc_out_temp[5] <= adc_in;
            end
            else if (count == 7'd87) begin
               acc_out_temp[4] <= adc_in;
            end
            else if (count == 7'd89) begin
               acc_out_temp[3] <= adc_in;
            end
            else if (count == 7'd91) begin
               acc_out_temp[2] <= adc_in;
            end
            else if (count == 7'd93) begin
               acc_out_temp[1] <= adc_in;
            end
            else if (count == 7'd95) begin
               acc_out_temp[0] <= adc_in;
            end
            else begin end

            count <= count + 1;
      end
      else if (count == 7'd127) begin
         start <= 1'b0;
            count <= 1'b0;
      end
      else begin
            count <= count + 1;
            end
        end



endmodule

// Gary Matthias
// Calculates the one-second peak
module acceleration_peak(reset, clock, accel_in, accel_peak_out);

   input reset, clock;
      wire reset, clock;

      input[9:0] accel_in;
      wire[9:0] accel_in;

      output[9:0] accel_peak_out;
      reg[9:0] accel_peak_out;

      reg[25:0] count;

      always @(posedge clock) begin
         if (reset) begin
               count <= 27'd0;
                  accel_peak_out <= 27'd0;
            end
            else if (accel_in > accel_peak_out) begin
               count <= 27'd0;
                  accel_peak_out <= accel_in;
            end
            else if (count == 27'd26999999) begin
               count <= 27'd0;
                  accel_peak_out <= 27'd0;
            end
            else begin
               count <= count + 1;
            end
      end

endmodule

////////////////////////////////////////////////////////////////////////////////////
///////
//
// Module Name:   d0606_6
// Engineer:      Anthony J. Quivers
//
//
// Decription:
// This module controls the occurence and reactions to all physical
// events envolved with game when the balls are set to motion.  Here
// are 16 FSM and registering systems used to store and manipulate the
// physical state variables of each ball. On table interactions such as
// collisions with the padding and other balls are modeled here.  The state
// of the balls are manipulated in a manner to simulate reality and capture
// the physical effects of the interactions.  This module outputs the state
// busses of each ball so that the user may be prompted for an input to initiate
// the gaming environment.
```

```
/////////////////////////////////////////////////////////////////////////////////

module physics_interface (clk, reset, write_bus, wball, wena,
                          ball0, ball1, ball2, ball3,
                          ball4, ball5, ball6, ball7,
                          ball8, ball9, ball10, ball11,
                          ball12, ball13, ball14, ball15)


input                      clk;
input                      reset;
input              [37:0]  write_bus;
input              [3:0]   wball;
input                      wena;

output             [37:0]  ball0;
output             [37:0]  ball1;
output             [37:0]  ball2;
output             [37:0]  ball3;
output             [37:0]  ball4;
output             [37:0]  ball5;
output             [37:0]  ball6;
output             [37:0]  ball7;
output             [37:0]  ball8;
output             [37:0]  ball9;
output             [37:0]  ball10;
output             [37:0]  ball11;
output             [37:0]  ball12;
output             [37:0]  ball13;
output             [37:0]  ball14;
output             [37:0]  ball15;

// Initial Ball Positions
parameter      [10:0]  ba0_xi = 600;
parameter      [10:0]  ba0_yi = 500;

parameter      [10:0]  ba1_xi = 1380;
parameter      [10:0]  ba1_yi = 500;

parameter      [10:0]  ba2_xi = 1440;
parameter      [10:0]  ba2_yi = 465;

parameter      [10:0]  ba3_xi = 1500;
parameter      [10:0]  ba3_yi = 430;

parameter      [10:0]  ba4_xi = 1560;
parameter      [10:0]  ba4_yi = 395

parameter      [10:0]  ba5_xi = 1620;
parameter      [10:0]  ba5_yi = 360;

parameter      [10:0]  ba6_xi = 1620;
parameter      [10:0]  ba6_yi = 430;

parameter      [10:0]  ba7_xi = 1560;
parameter      [10:0]  ba7_yi = 465;

parameter      [10:0]  ba8_xi = 1500;
parameter      [10:0]  ba8_yi = 500;
parameter       [10:0]  ba9_xi = 1440;
parameter       [10:0]  ba9_yi = 535;

parameter       [10:0]  ba10_xi = 1500;
parameter       [10:0]  ba10_yi = 570;

parameter       [10:0]  ba11_xi = 1560;
parameter       [10:0]  ba11_yi = 535;

parameter       [10:0]  ba12_xi = 1620;
parameter       [10:0]  ba12_yi = 500;

parameter       [10:0]  ba13_xi = 1620;
parameter       [10:0]  ba13_yi = 570;

parameter       [10:0]  ba14_xi = 1560;
parameter       [10:0]  ba14_yi = 605;

parameter       [10:0]  ba15_xi = 1620;
parameter       [10:0]  ba15_yi = 640;
```

```
//////////////////////////////////////////////////////////
//                                                      //
// -- Clock Systems and Timing Modules --               //
//                                                      //
//                                                      //
//////////////////////////////////////////////////////////
// Clock Systems and Timeing Modules
wire                    cyc;
wire                    cyc_coll;
wire                    ret;
wire                    bit4;
wire                    bit3;
wire                    bit2;
wire                    bit1;
wire                    bit0;

cyclecounter cyc_gen(clk, reset, cyc);
framecounter ret_gen(clk, reset, ret);
cyc_shifter  cyc_2_gen(clk, cyc, reset, cyc_coll);
cyc_vel_bit4 bit4_gen(cyc,reset, bit4);
cyc_vel_bit3 bit3_gen(cyc,reset, bit3);
cyc_vel_bit2 bit2_gen(cyc,reset, bit2);
cyc_vel_bit1 bit1_gen(cyc,reset, bit1);
cyc_vel_bit0 bit0_gen(cyc,reset, bit0);
```

```
//////////////////////////////////////////////////////////////
//                                                          //
// -- Ball State Registers and FSMs --                      //
//                                                          //
//                                                          //
//////////////////////////////////////////////////////////////

//  Ball State Register and Positional FSMs
wire            [37:0]  ball0_in;
wire            [37:0]  ball0_out;
wire            [37:0]  ball1_in;
wire            [37:0]  ball1_out;
wire            [37:0]  ball2_in;
wire            [37:0]  ball2_out;
wire            [37:0]  ball3_in;
wire            [37:0]  ball3_out;
wire            [37:0]  ball4_in;
wire            [37:0]  ball4_out;
wire            [37:0]  ball5_in;
wire            [37:0]  ball5_out;
wire            [37:0]  ball6_in;
wire            [37:0]  ball6_out;
wire            [37:0]  ball7_in;
wire            [37:0]  ball7_out;
wire            [37:0]  ball8_in;
wire            [37:0]  ball8_out;
wire            [37:0]  ball9_in;
wire            [37:0]  ball9_out;
wire            [37:0]  ball10_in;
wire            [37:0]  ball10_out;
wire            [37:0]  ball11_in;
wire            [37:0]  ball11_out;
wire            [37:0]  ball12_in;
wire            [37:0]  ball12_out;
wire            [37:0]  ball13_in;
wire            [37:0]  ball13_out;
wire            [37:0]  ball14_in;
wire            [37:0]  ball14_out;
wire            [37:0]  ball15_in;
wire            [37:0]  ball15_out;
wire            [37:0]  coll_bus;
wire            [3:0]   sel_a;
wire                    coll_ena_f;
wire                    vel_write;

// Ball 0
ball_state_reg  ball0_state(clk, reset, 0, write_bus, wball, wena,
                            coll_bus, sel_a, coll_ena_f,
                            1, 1, 0, 0, 0, 0, ba0_yi, ba0_xi,
                            ball0_in);
ball_state_fsm  ball0_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball0_in,
                            ball0_out);

// Ball 1
ball_state_reg  ball1_state(clk, reset, 1, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba1_yi, ba1_xi,
                             ball1_in);
ball_state_fsm  ball1_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball1_in,
                             ball1_out);

// Ball 2
ball_state_reg  ball2_state(clk, reset, 2, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba2_yi, ba2_xi,
                             ball2_in);
ball_state_fsm  ball2_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball2_in,
                             ball2_out);

// Ball 3
ball_state_reg  ball3_state(clk, reset, 3, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba3_yi, ba3_xi,
                             ball3_in);
ball_state_fsm  ball3_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball3_in,
                             ball3_out);

// Ball 4
ball_state_reg  ball4_state(clk, reset, 4, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba4_yi, ba4_xi,
                             ball4_in);
ball_state_fsm  ball4_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball4_in,
                             ball4_out);

// Ball 5
ball_state_reg  ball5_state(clk, reset, 5, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba5_yi, ba5_xi,
                             ball5_in);
ball_state_fsm  ball5_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball5_in,
                             ball5_out);

// Ball 6
ball_state_reg  ball6_state(clk, reset, 6, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba6_yi, ba6_xi,
                             ball6_in);
ball_state_fsm  ball6_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball6_in,
                             ball6_out);

// Ball 7
ball_state_reg  ball7_state(clk, reset, 7, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba7_yi, ba7_xi,
                             ball7_in);
ball_state_fsm  ball7_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball7_in,
                             ball7_out);

// Ball 8
ball_state_reg  ball8_state(clk, reset, 8, write_bus, wball, wena,
                             coll_bus, sel_a, coll_ena_f,
                             1, 1, 0, 0, 0, 0, ba8_yi, ba8_xi,
                             ball8_in);
```

```
ball_state_fsm  ball8_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball8_in,
                          ball8_out);

// Ball 9
ball_state_reg  ball9_state(clk, reset, 9, write_bus, wball, wena,
                          coll_bus, sel_a, coll_ena_f,
                          1, 1, 0, 0, 0, 0, ba9_yi, ba9_xi,
                          ball9_in);
ball_state_fsm  ball9_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball9_in,
                          ball9_out);

// Ball 10
ball_state_reg  ball10_state(clk, reset, 10, write_bus, wball, wena,
                          coll_bus, sel_a, coll_ena_f,
                          1, 1, 0, 0, 0, 0, ba10_yi, ba10_xi,
                          ball10_in);
ball_state_fsm  ball10_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball10_in,
                          ball10_out);

// Ball 11
ball_state_reg  ball11_state(clk, reset, 11, write_bus, wball, wena,
                          coll_bus, sel_a, coll_ena_f,
                          1, 1, 0, 0, 0, 0, ba11_yi, ba11_xi,
                          ball11_in);
ball_state_fsm  ball11_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball11_in,
                          ball11_out);

// Ball 12
ball_state_reg  ball12_state(clk, reset, 12, write_bus, wball, wena,
                          coll_bus, sel_a, coll_ena_f,
                          1, 1, 0, 0, 0, 0, ba12_yi, ba12_xi,
                          ball12_in);
ball_state_fsm  ball12_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball12_in,
                          ball12_out);

// Ball 13
ball_state_reg  ball13_state(clk, reset, 13, write_bus, wball, wena,
                          coll_bus, sel_a, coll_ena_f,
                          1, 1, 0, 0, 0, 0, ba13_yi, ba13_xi,
                          ball13_in);
ball_state_fsm  ball13_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball13_in,
                          ball13_out);

// Ball 14
ball_state_reg  ball14_state(clk, reset, 14, write_bus, wball, wena,
                          coll_bus, sel_a, coll_ena_f,
                          1, 1, 0, 0, 0, 0, ba14_yi, ba14_xi,
                          ball14_in);
ball_state_fsm  ball14_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball14_in,
                          ball14_out);

// Ball 15
ball_state_reg  ball15_state(clk, reset, 15, write_bus, wball, wena,
                          coll_bus, sel_a, coll_ena_f,
                          1, 1, 0, 0, 0, 0, ba15_yi, ba15_xi,
                          ball15_in);
ball_state_fsm  ball15_fsm(clk, cyc, ret, bit4, bit3, bit2, bit1, bit0, ball15_in,
                          ball15_out);
```

```
// Flush Module
flush input_sheild(clk, cyc, reset, wena, vel_write, coll_ena_f);

//  Continuous Output Assignments
assign ball0 = ball0_in;
assign ball1 = ball1_in;
assign ball2 = ball2_in;
assign ball3 = ball3_in;
assign ball4 = ball4_in;
assign ball5 = ball5_in;
assign ball6 = ball6_in;
assign ball7 = ball7_in;
assign ball8 = ball8_in;
assign ball9 = ball9_in;
assign ball10 = ball10_in;
assign ball11 = ball11_in;
assign ball12 = ball12_in;
assign ball13 = ball13_in;
assign ball14 = ball14_in;
assign ball15 = ball15_in;

//  Mutiplexor Structures for Collision FSM
wire            [37:0]  state_a;
wire            [37:0]  state_b;
wire            [3:0]   sel_b;

mux_16_1 mux_a(ball0_out, ball1_out, ball2_out, ball3_out, ball4_out,
                ball5_out, ball6_out, ball7_out, ball8_out, ball9_out,
                ball10_out, ball11_out, ball12_out, ball13_out, ball14_out,
                ball5_out, sel_a, state_a);

mux_16_1 mux_b(ball0_out, ball1_out, ball2_out, ball3_out, ball4_out,
                ball5_out, ball6_out, ball7_out, ball8_out, ball9_out,
                ball10_out, ball11_out, ball12_out, ball13_out, ball14_out,
                ball5_out, sel_b, state_b);

// Physical Interation Managers
wire                [10:0]  rx_a;
wire                [10:0]  ry_a;
wire                [5:0]   vx_a_init;
wire                        vxs_a_init;
wire                [5:0]   vy_a_init;
wire                        vys_a_init;
wire                        active_a;
wire                        still_a;

wire                [10:0]  rx_b;
wire                [10:0]  ry_b;
wire                [5:0]   vx_b_init;
wire                        vxs_b_init;
wire                [5:0]   vy_b_init;
wire                        vys_b_init;
wire                        active_b;
wire                        still_b;

// Input State Bus Splitters
splitter state_a_split(state_a, active_a, still_a, vys_a_init, vy_a_init,
vxs_a_init, vx_a_init, ry_a, rx_a);
splitter state_b_split(state_b, active_b, still_b, vys_b_init, vy_b_init,
vxs_b_init, vx_b_init, ry_b, rx_b);







////////////////////////////////////////////////////////////
//                                                        //
// -- Collision Manager Region --                         //
//                                                        //
//                                                        //
////////////////////////////////////////////////////////////
// FSM Control Signals
wire                    coll_trigo;
wire                    wall_trigo;
wire                    v_a_reg_inito;
wire                    v_b_reg_inito;
wire                    v_a_reg_wallo;
wire                    v_a_reg_collo;
wire                    inc_ao;
wire                    inc_bo;
wire                    vel_writeo;

wire                    coll_trig;
wire                    wall_trig;
wire                    v_a_reg_init;
wire                    v_b_reg_init;
wire                    v_a_reg_wall;
wire                    v_a_reg_coll;
wire                    inc_a;
wire                    inc_b;

// Control Signal Regiisters
control_sig_reg deglitcher (clk, coll_trigo, wall_trigo, v_a_reg_inito,
v_b_reg_inito, v_a_reg_wallo, v_a_reg_collo, inc_ao, inc_bo, vel_writeo,
                            coll_trig, wall_trig, v_a_reg_init, v_b_reg_init,
v_a_reg_wall, v_a_reg_coll, inc_a, inc_b, vel_write);

// Registered State for Ball A
wire            [5:0]   vx_a;
wire                    vxs_a;
wire            [5:0]   vy_a;
wire                    vys_a;

// Registered State for Ball B
wire            [5:0]   vx_b;
wire                    vxs_b;
wire            [5:0]   vy_b;
wire                    vys_b;

// Inputs From Wall  Dyamics
wire            [5:0]   vx_wall;
wire                    vxs_wall;
wire            [5:0]   vy_wall;
wire                    vys_wall;

wall_dynamics rebound (wall_trig, vx_a, vxs_a, vy_a, vys_a, rx_a, ry_a,
                              vx_wall, vxs_wall, vy_wall, vys_wall);

// Inputs From Collision Dyamics
wire            [5:0]   vx_coll;
wire                    vxs_coll;
wire            [5:0]   vy_coll;
wire                    vys_coll;

collision_dynamics recochet(coll_trig, vx_a, vxs_a, vy_a, vys_a, rx_a, ry_a
                                vx_b, vxs_b, vy_b, vys_b, rx_b, ry_b
                                vx_coll, vxs_coll, vy_coll, vys_coll);

// Main Collision Manager
wire            [10:0]  x_a;
wire            [10:0]  y_a;
wire            [10:0]  x_b;
wire            [10:0]  y_b;
wire                    still;
wire                    active;

collision_fsm collision_mgr(   clk, cyc_coll, reset,
       rx_a, ry_a, vx_a_init, vxs_a_init, vy_a_init, vys_a_init, still_a, active_a,
       rx_b, ry_b, vx_b_init, vxs_b_init, vy_b_init, vys_b_init, still_b, active_b,
       vx_coll, vxs_coll, vy_coll, vys_coll, vx_wall, vxs_wall, vy_wall, vys_wall,
       v_a_reg_init, v_b_reg_init, v_a_reg_wall, v_a_reg_coll, inc_a, inc_b,
       coll_trigo, wall_trigo, v_a_reg_inito, v_b_reg_inito, v_a_reg_wallo,
       v_a_reg_collo, inc_ao, inc_bo, vel_writeo,
       sel_a, sel_b, vx_a, vxs_a, vy_a, vys_a,
       vx_b, vxs_b, vy_b, vys_b, x_a, y_a, x_b, y_b, still, active);

// The Collision Manger Write Bus Condensor: The Final Touch!
condenser collision_mgr_bus(active, still, vys_a, vy_a, vxs_a, vx_a, y_a, x_a,
coll_bus);
```

GET_INTERSECTIONS

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17:46:50 05/06/2007
// Design Name:
// Module Name: Get_Intersections
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module Get_Intersections(bx1,bx2,bx3,bx4,bx5,bx6,bx7,bx8,bx9,bx10,bx11,
bx12,bx13,bx14,bx15,
by1,by2,by3,by4,by5,by6,by7,by8,by9,by10,by11,by12,by13,by14,by15,
bp1,bp2,bp3,bp4,bp5,bp6,bp7,bp8,bp9,bp10,bp11,bp12,bp13,bp14,bp15,
tx,ty,px,py,
player,
E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2,
clk,
finalin,finalobject,
xf,yf,zf,nxf,nyf,nzf
);
output reg [2:0]finalin;
output reg [7:0] finalobject;
output reg signed [15:0]xf,yf,zf,nxf,nyf,nzf;
input player;
input bp1,bp2,bp3,bp4,bp5,bp6,bp7,bp8,bp9,bp10,bp11,bp12,bp13,bp14,bp15;
input clk;
input signed [15:0] E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2;
input signed
[15:0]bx1,bx2,bx3,bx4,bx5,bx6,bx7,bx8,bx9,bx10,bx11,bx12,bx13,bx14,bx15;
input signed
[15:0]by1,by2,by3,by4,by5,by6,by7,by8,by9,by10,by11,by12,by13,by14,by15;
input signed [15:0]tx,ty,px,py;
reg signed[15:0] x0=4000;
reg signed[15:0] y0=4000;
reg signed[15:0] tpz=1000;
//get cue intersection
reg [7:0]cueobject;
wire [7:0]cueobjectw;
wire signed [15:0]cuexw,cueyw,cuezw,cuenxw,cuenyw,cuenzw;
reg signed[15:0] cuex,cuey,cuez,cuenx,cueny,cuenz;
reg [2:0]cuepin;
wire [2:0]cuepinw;
Get_Cue_Intersections GCI1(.tx(tx),.ty(ty),.px(px),.py(py),.player(player),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.objectc(cueobjectw),
.Pxc(cuexw),.Pyc(cueyw),.Pzc(cuezw),
.PNVxc(cuenxw),.PNVyc(cuenyw),.PNVzc(cuenzw),
.Pin(cuepinw),.tpz(tpz),.clk(clk));
//get table intersection
wire signed [15:0]tablexw,tableyw,tablezw,tablenxw,tablenyw,tablenzw;
reg signed[15:0] tablex,tabley,tablez,tablenx,tableny,tablenz;
reg [2:0]tablein;
reg [7:0] tableobject;
wire [2:0]tableinw;
wire [7:0] tableobjectw;
Get_Plane_Intersections GPID(.x(tablexw),.y(tableyw),.z(tablezw),.Pin(tablepinw),
.pNVx(tablenxw),.pNVy(tablenyw),.pNVz(tablenzw),
.
E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2),
.E_P_Lz2(E_P_Lz2),.object(tableobjectw),.clk(clk));
//get intersection with balls
wire signed [15:0]ballxw,ballyw,ballzw,ballxnvw,ballynvw,ballznvw;
reg signed[15:0] ballx,bally,ballz,ballxnv,ballynv,ballznv;
reg [7:0]ballobject;
reg [2:0]ballin;
wire [7:0]ballobjectw;
wire [2:0]ballinw;
Get_Ball_Intersections GBI1(.bin(ballinw), .px(ballxw), .py(ballyw), .pz(ballzw),
.pxnv(ballxnvw),
.pynv(ballnyvw), .pznv(ballnzvw), .object(ballobjectw),
.bx1(bx1), .bx2(bx2), .bx3(bx3), .bx4(bx4), .bx5(bx5), .bx6(bx6), .bx7(bx7),
.bx8(bx8), .bx9(bx9),
.bx10(bx10), .bx11(bx11), .bx12(bx12), .bx13(bx13), .bx14(bx14), .bx15(bx15),
.by1(by1), .by2(by2), .by3(by3), .by4(by4), .by5(by5), .by6(by6), .by7(by7),
.by8(by8), .by9(by9),
.by10(by10), .by11(by11), .by12(by12), .by13(by13), .by14(by14), .by15(by15),
.bp1(bp1), .bp2(bp2), .bp3(bp3), .bp4(bp4), .bp5(bp5), .bp6(bp6), .bp7(bp7),
.bp8(bp8), .bp9(bp9),
.bp10(bp10), .bp11(bp11), .bp12(bp12), .bp13(bp13), .bp14(bp14), .bp15(bp15),
.E_P_Lx(E_P_Lx), . E_P_Ly(E_P_Ly), . E_P_Lz(E_P_Lz),
. E_P_Lx2(E_P_Lx2), . E_P_Ly2(E_P_Ly2), . E_P_Lz2(E_P_Lz2),
.clk(clk)
);
//which is the closest?
wire signed [15:0]xfw,yfw,zfw;
reg signed[15:0] zero =0;
reg [2:0]fin;
wire[2:0]finw;
wire [2:0]one=1;
Closest
C3(.p1x(ballxw),.p1y(ballyw),.p1z(ballzw),.p2x(tablexw),.p2y(tableyw),.p2z(tablezw)
,
.p3x(cuexw),.p3y(cueyw),.p3z(cuezw),.p4x(zero),.p4y(zero),.p4z(zero),.p5x(zero),.p5
y(zero),.p5z(zero
),.p6x(zero),.p6y(zero),.p6z(zero),
.P1in(ballin), .P2in(/*tablein*/one), .P3in(/*cuein*/one), .P4in(one), .P5in(one),
.P6in(one), .Pin(finw),
.Pxc(xfw),.Pyc(yfw),.Pzc(zfw),
.clk(clk)
);
always@(posedge clk)
begin
xf=xfw;yf=yfw;zf=zfw;
cuex=cuexw;cuey=cueyw;cuez=cuezw;
cuenx=cuenxw;cueny=cuenyw;cuenz=cuenzw;
tablex=tablexw; tabley=tableyw; tablez=tablezw;
tablenx=tablenxw; tableny=tablenyw; tablenz=tablenzw;
ballx=ballxw; bally=ballyw; ballz=ballzw;
ballxnv=ballxnvw; ballynv=ballynvw; ballznv=ballznvw;
cuepin=cuepinw;cueobject=cueobjectw;
tablein=tableinw;tableobject=tableobjectw;
ballobject=ballobjectw;ballin=ballinw;
fin=finw;
if((finalin!=1)&&(finalin!=2))begin finalin=4;end
if(fin==1)finalin=1;
else if(fin==2)
begin
finalin=2;
if((xf==ballx)&&(yf==bally)&&(zf==ballz))begin nxf=ballxnv;nyf=ballynv;nzf=ballznv;
finalobject=ballobject;end
if((xf==cuex)&&(yf==cuey)&&(zf==cuez))begin
nxf=cuenx;nyf=cueny;nzf=cuenz;finalobject=cueobject;end
if((xf==tablex)&&(yf==tabley)&&(zf==tablez))begin
nxf=tablenx;nyf=tableny;nzf=tablenz;finalobject=tableobject;end
end
end
endmodule
```

GET_CUE INTERSECTIONS

```
`timescale 1ns / 1ps
module Get_Cue_Intersections(tx,ty,px,py,player,
E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2,
objectc,
Pxc,Pyc,Pzc,
PNVxc,PNVyc,PNVzc,
Pin,tpz,clk);
output wire signed [15:0]Pxc,Pyc,Pzc,PNVxc,PNVyc,PNVzc;
output reg [2:0]Pin;
output wire [7:0]objectc;
wire [2:0]Pinw;
input clk;
input signed [15:0]tx,ty,px,py;
reg signed[15:0] tx2,ty2,px2,py2;
input wire player;
input signed [15:0] E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2,tpz;
reg signed[15:0] CueWidth = 30;
reg signed[15:0] CueLength = 1000;
reg signed[15:0] dx,Dx,dy,Dy,dh,sino,coso;
wire [15:0]tx2w,ty2w,px2w,py2w;
wire [15:0]tpzw;
wire [15:0]tpzpc;
wire [15:0]tpzmc;
assign tx2w=tx2;assign ty2w=ty2;assign px2w=px2;assign py2w=py2;
assign tpzw=tpz;
assign tpzpc=tpz+CueWidth;
assign tpzmc=tpz-CueWidth;
reg [2:0]P1in, P2in, P3in, P4in, P5in, P6in;
reg [7:0]object;
reg[3:0]dimc,dimc2;
wire [3:0]dimcw,dimc2w;
assign dimcw=dimc;assign dimc2w=dimc2;
wire [3:0]zero;
wire [3:0]one;
wire [3:0]two;
assign one=1;
assign two=2;
assign zero=0;
Cube_Intersections CI1(.Pxct(Pxc),.Pyct(Pyc),.Pzct(Pzc),
.
PNVxct(PNVxc),.PNVyct(PNVyc),.PNVzct(PNVzc),.Pint(Pinw),.objectt(objectc),
.
E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.
dim1(three), .dim2(three), .dim3(dimc2w), .dim4(dimc2w), .dim5(dimc1w),
.dim6(dimc1w),
.
object1(object), .object2(object), .object3(object), .object4(object),
.
object5(object), .object6(object),
.
PL1x1(txw),.PL1y1(tyw),.PL1z1(tpzpc),
.
PL1x2(tx2w),.PL1y2(ty2w),.PL1z2(tpzpc),
.
PL1x3(pxw),.PL1y3(pyw),.PL1z3(tpzpc),
.
PL2x1(txw),.PL2y1(tyw),.PL2z1(tpzw),
.
PL2x2(tx2w),.PL2y2(ty2w),.PL2z2(tpzw),
.
PL2x3(pxw),.PL2y3(pyw),.PL2z3(tpzw),
.
PL3x1(txw),.PL3y1(tyw),.PL3z1(tpzw),
.
PL3x2(txw),.PL3y2(tyw),.PL3z2(tpzpc),
.
PL3x3(pxw),.PL3y3(pyw),.PL3z3(tpzw),
.
PL4x1(tx2w),.PL4y1(ty2w),.PL4z1(tpzw),
.
PL4x2(tx2w),.PL4y2(ty2w),.PL4z2(tpzpc),
.
PL4x3(px2w),.PL4y3(py2w),.PL4z3(tpzw),
.
PL5x1(pxw),.PL5y1(pyw),.PL5z1(tpzpc),
.
PL5x2(pxw),.PL5y2(pyw),.PL5z2(tpzw),
.
PL5x3(px2w),.PL5y3(py2w),.PL5z3(tpzw),
.
PL6x1(txw),.PL6y1(tyw),.PL6z1(tpzpc),
.
PL6x2(txw),.PL6y2(tyw),.PL6z2(tpzw),
.
PL6x3(tx2w),.PL6y3(ty2w),.PL6z3(tpzpc),.clk(clk)
);
always@(posedge clk)
begin
Pin=Pinw;
if((Pin!=1)&&(Pin!=2))begin Pin=4;end
dimc=1;dimc=2;
if (player) object = 3;
else object = 4;
if(tx-px==0)dimc=1;dimc2=2;
if(ty-py==0)dimc=2;dimc2=1;
if (tx-px!=0)
begin
dx = px-tx;
if(dx<0)dx = -dx;
dy = py-ty;
if(dy<0)dy = - dy;
dh = CueLength+24;//approx
coso = (1024*dy);
sino = (1024*dx);
if( (1000*(ty-py))>0)
begin
Dy=((CueWidth*sino)/1024); //mm
Dx=((CueWidth*coso)/1024); //mm
tx2=tx-Dx;
ty2=ty+Dy;
px2=px-Dx;
py2=py+Dy;
end
else
begin
```

```
Dx=((CueWidth*sino)/1024); //mm
Dy=((CueWidth*coso)/1024); //mm
tx2=tx-Dx;
ty2=ty-Dy;
px2=px-Dx;
py2=py-Dy;
end
end
else
begin
tx2=tx-CueWidth;
ty2=ty;
px2 = px-CueWidth;
py2 = py;
end
end
endmodule
```

VGA

```
// This module provides control signals to the ADV7125
// such that the resolution is 640x480 and the refresh
// rate is 75Hz.
// hsync is active low: high for 640 pixels of active video,
// high for 16 pixels of front porch,
// low for 96 pixels of hsync,
// high for 48 pixels of back porch
// vsync is active low: high for 480 lines of active video,
// high for 11 lines of front porch,
// low for 2 lines of vsync,
// high for 32 lines of back porch
module vga (pixel_clock, reset, hsync, vsync, sync_b,
blank_b, pixel_count, line_count);
input pixel_clock; // 31.5 MHz pixel clock
input reset; // system reset
output hsync; // horizontal sync
output vsync; // vertical sync
output sync_b; // hardwired to Vdd
output blank_b; // composite blank
output [9:0] pixel_count; // number of the current pixel
output [9:0] line_count; // number of the current line
// 640x480 75Hz parameters
parameter PIXELS = 800;
parameter LINES = 525;
parameter HACTIVE_VIDEO = 640;
parameter HFRONT_PORCH = 16;
parameter HSYNC_PERIOD = 96;
parameter HBACK_PORCH = 48;
parameter VACTIVE_VIDEO = 480;
parameter VFRONT_PORCH = 11;
parameter VSYNC_PERIOD = 2;
parameter VBACK_PORCH = 32;
// current pixel count
reg [9:0] pixel_count = 10'b0;
reg [9:0] line_count = 10'b0;
// registered outputs
reg hsync = 1'b1;
reg vsync = 1'b1;
reg blank_b = 1'b1;
wire sync_b; // connected to Vdd
wire pixel_clock;
wire [9:0] next_pixel_count;
wire [9:0] next_line_count;
always @ (posedge pixel_clock)
begin
if (reset)
begin
pixel_count <= 10'b0;
line_count <= 10'b0;
hsync <= 1'b1;
vsync <= 1'b1;
blank_b <= 1'b1;
end
else
begin
pixel_count <= next_pixel_count;
line_count <= next_line_count;
hsync <=
(next_pixel_count < HACTIVE_VIDEO + HFRONT_PORCH) |
(next_pixel_count >= HACTIVE_VIDEO+HFRONT_PORCH+
HSYNC_PERIOD);
vsync <=
(next_line_count < VACTIVE_VIDEO+VFRONT_PORCH) |
(next_line_count >= VACTIVE_VIDEO+VFRONT_PORCH+
VSYNC_PERIOD);
// this is the and of hblank and vblank
blank_b <=
(next_pixel_count < HACTIVE_VIDEO) &
(next_line_count < VACTIVE_VIDEO);
end
end
// next state is computed with combinational logic
assign next_pixel_count = (pixel_count == PIXELS-1) ?
10'h000 : pixel_count + 1'b1;
assign next_line_count = (pixel_count == PIXELS-1) ?
(line_count == LINES-1) ? 10'h000 :
line_count + 1'b1 : line_count;
// since we are providing hsync and vsync to the display, we
// can hardwire composite sync to Vdd.
assign sync_b = 1'b1;
endmodule
```

PIXEL COLOR

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 15:37:10 05/10/2007
// Design Name:
// Module Name: pixelcolor
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module pixelcolor(pixel_count,line_count,reset,
bx1,bx2,bx3,bx4,bx5,bx6,bx7,bx8,bx9,bx10,bx11,bx12,bx13,bx14,bx15,
by1,by2,by3,by4,by5,by6,by7,by8,by9,by10,by11,by12,by13,by14,by15,
bp1,bp2,bp3,bp4,bp5,bp6,bp7,bp8,bp9,bp10,bp11,bp12,bp13,bp14,bp15,
tx,ty,px,py,
player,
clk2,ram1dataw,ram2dataw,ram1add,ram2add,write_b_ram1,write_b_ram2,RGBout
);
input reset;
input player,clk2;
input bp1,bp2,bp3,bp4,bp5,bp6,bp7,bp8,bp9,bp10,bp11,bp12,bp13,bp14,bp15;
input [15:0] bx1,bx2,bx3,bx4,bx5,bx6,bx7,bx8,bx9,bx10,bx11,bx12,bx13,bx14,bx15;
input [15:0] by1,by2,by3,by4,by5,by6,by7,by8,by9,by10,by11,by12,by13,by14,by15;
input [15:0] tx,ty,px,py;
input [10:0] pixel_count,line_count;
inout [35:0]ram1dataw,ram2dataw;
output reg [23:0]RGBout;
output reg [18:0]ram1add,ram2add;
output reg write_b_ram1,write_b_ram2;
reg check;
reg dataready;
reg state;
reg [6:0]glum=60;
reg [1:0]counter;
reg [2:0]temp1in,temp2in;
wire [2:0]temp1inw,temp2inw;
reg [3:0]temp1object,temp2object;
reg [6:0]lum,object;
reg [7:0]hue,sat;
reg [10:0] mypixel_count,myline_count;
reg [23:0]RGB;
reg [23:0]ram1data,ram2data;
reg signed[15:0] tempx1,tempy1,tempz1,tempnx1,tempny1,tempnz1;
reg signed[15:0] tempx2,tempy2,tempz2,tempnx2,tempny2,tempnz2;
reg signed[15:0] light=4000;
reg signed[15:0] den1,den2,den1sqd,den2sqd;
reg signed[15:0] x0=2;
reg signed[15:0] y0=2;
////////////////////////////////////////////
//divide stuff
reg signed[15:0] quotient1;
wire [15:0]quotient1w;
wire [15:0]divisor1w;
wire [15:0]dividend1w;
wire [15:0]remainder1w;
wire rfd1w;
reg signed[15:0] quotient2;
wire [15:0]quotient2w;
wire [15:0]divisor2w;
wire [15:0]dividend2w;
wire [15:0]remainder2w;
wire rfd2w;
reg signed[15:0] quotient3;
wire [15:0]quotient3w;
wire [15:0]divisor3w;
wire [15:0]dividend3w;
wire [15:0]remainder3w;
wire rfd3w;
/////////////////////////////////////////////
wire signed [15:0]tempx1pre,tempy1pre,tempz1pre,tempnx1pre,tempny1pre,tempnz1pre;
wire signed [15:0]tempx2pre,tempy2pre,tempz2pre,tempnx2pre,tempny2pre,tempnz2pre;
wire[15:0]den1sqdw,den2sqdw,den1w,den2w;
wire rdy1,rdy2;
wire signed [15:0]eplx,eplx2,eply,eply2,eplz,eplz2;
wire signed
[15:0]bx1,bx2,bx3,bx4,bx5,bx6,bx7,bx8,bx9,bx10,bx11,bx12,bx13,bx14,bx15;
wire signed
[15:0]by1,by2,by3,by4,by5,by6,by7,by8,by9,by10,by11,by12,by13,by14,by15;
wire signed [15:0]tx,ty,px,py;
reg signed[15:0] eplxpre,eplx2pre,eplypre,eply2pre,eplzpre,eplz2pre;
reg signed[15:0]
bx1pre,bx2pre,bx3pre,bx4pre,bx5pre,bx6pre,bx7pre,bx8pre,bx9pre,bx10pre,bx11pre,bx12
pre,bx13pre,
bx14pre,bx15pre;
reg signed[15:0]
by1pre,by2pre,by3pre,by4pre,by5pre,by6pre,by7pre,by8pre,by9pre,by10pre,by11pre,by12
pre,by13pre,
by14pre,by15pre;
reg signed[15:0] txpre,typre,pxpre,pypre;
parameter state1=0;
parameter state2=1;
always@(posedge clk2)
begin
if(counter==0)
begin
dataready=((temp1in==1)|(temp1in==2))&&((temp2in==1)|(temp2in==2))&&(rdy1 && rdy2);
end
if(dataready&&counter==0)
begin
mypixel_count = (mypixel_count == 800-1) ? 10'h000 : mypixel_count + 1'b1;
myline_count = (mypixel_count == 800-1) ? (myline_count == 525-1)
? 10'h000 : myline_count + 1'b1 : myline_count;
end
if(counter<2)begin counter<=counter+1;end
if(counter==2)begin counter<=0;end
if(reset)
begin
counter<=0;
check<=0;
mypixel_count=0;
myline_count=0;
end
case(state)
state1://writing to ram1, reading from ram2
begin
RGBout=ram2dataw;
if(mypixel_count==1)begin state=state2;end
if(dataready&&counter==0) begin
ram1add=((myline_count*799)+pixel_count);check<=1;end
if(dataready&&counter==2&&check==1)begin ram1data=RGB;
write_b_ram1=0;check<=0; end
if(counter==0)begin ram2add=((line_count*799)+pixel_count);end
if(counter==2)begin ram2add=((line_count*799)+pixel_count);end
end
//0b110011000110111010
```

```
state2://writing to ram2, reading from ram1
begin
RGBout=ram1dataw;
if(mypixel_count==1)begin state=state2;end
if(dataready&&counter==0)begin
ram2add=((myline_count*799)+mypixel_count);check<=1;end
if(dataready&&counter==2&&check==1)begin ram2data=RGB;
write_b_ram2=0;check<=0;end
if(counter==0)begin ram1add=((line_count*799)+pixel_count);end
if(counter==2)begin ram1add=((line_count*799)+pixel_count);end
end
endcase
end//
assign ram1dataw=ram1data;
assign ram2dataw=ram2data;
//feed values to Eye_Plane_Line
Eye_Plane_Line EPL1(.E_P_Lx(eplx), .E_P_Ly(eply), .E_P_Lz(eplz),
.E_P_Lx2(eplx2), .E_P_Ly2(eply2), .E_P_Lz2(eplz2),.clk(clk2),
.pixel_count(mypixel_count),.line_count(myline_count));
//get intersections
Get_Intersections GI1(
.bx1(bx1),.bx2(bx2),.bx3(bx3),.bx4(bx4),.bx5(bx5),.bx6(bx6),.bx7(bx7),.bx8(bx8),.bx
9(bx9),.bx10(bx
10),.bx11(bx11),
.bx12(bx12),.bx13(bx13),.bx14(bx14),.bx15(bx15),
.by1(by1),.by2(by2),.by3(by3),.by4(by4),.by5(by5),.by6(by6),.by7(by7),.by8(by8),.by
9(by9),.by10(by
10),.by11(by11),
.by12(by12),.by13(by13),.by14(by14),.by15(by15),
.bp1(bp1),.bp2(bp2),.bp3(bp3),.bp4(bp4),.bp5(bp5),.bp6(bp6),.bp7(bp7),.bp8(bp8),.bp
9(bp9),.bp10(bp
10),.bp11(bp11),
.bp12(bp12),.bp13(bp13),.bp14(bp14),.bp15(bp15),
.tx(tx),.ty(ty),.px(px),.py(py),
.player(player),
.E_P_Lx(eplx), .E_P_Ly(eply), .E_P_Lz(eplz), .E_P_Lx2(eplx2), .E_P_Ly2(eply2),
.E_P_Lz2(eplz2),
.clk(clk2),
.finalin(temp1inw),.finalobject(temp1objectw),
.xf(tempx1pre),.yf(tempy1pre),.zf(tempz1pre),.nxf(tempnx1pre),.nyf(tempny1pre),.nzf
(tempnz1pre)
);
//get shadow
Get_Intersections GI2(
.bx1(bx1),.bx2(bx2),.bx3(bx3),.bx4(bx4),.bx5(bx5),.bx6(bx6),.bx7(bx7),.bx8(bx8),.bx
9(bx9),.bx10(bx
10),.bx11(bx11),
.bx12(bx12),.bx13(bx13),.bx14(bx14),.bx15(bx15),
.by1(by1),.by2(by2),.by3(by3),.by4(by4),.by5(by5),.by6(by6),.by7(by7),.by8(by8),.by
9(by9),.by10(by
10),.by11(by11),
.by12(by12),.by13(by13),.by14(by14),.by15(by15),
.bp1(bp1),.bp2(bp2),.bp3(bp3),.bp4(bp4),.bp5(bp5),.bp6(bp6),.bp7(bp7),.bp8(bp8),.bp
9(bp9),.bp10(bp
10),.bp11(bp11),
.bp12(bp12),.bp13(bp13),.bp14(bp14),.bp15(bp15),
.tx(tx),.ty(ty),.px(px),.py(py),
.player(player),
.E_P_Lx(light), .E_P_Ly(light), .E_P_Lz(light), .E_P_Lx2(tempx1), .E_P_Ly2(tempy1),
.E_P_Lz2(tempz1),
.clk(clk2),
.finalin(temp2inw),.finalobject(temp2objectw),
.xf(tempx2pre),.yf(tempy2pre),.zf(tempz2pre),.nxf(tempnx2pre),.nyf(tempny2pre),.nzf
(tempnz2pre)
);
//square root modules
assign den1sqdw=den1sqd;
assign den2sqdw=den2sqd;
assign dividend1w = ((((light-tempx1)*tempnx1)+
((light-tempy1)*tempny1)+
((light-tempz1)*tempnz1))*120);
assign divisor1w=den1*den2;
divide divider1(
.clk(clk2),
.dividend(dividend1w),
.divisor(divisor1w),
.quotient(quotient1w),
.remainder(remainder1w),
.rfd(rfd1w)
);
assign dividend2w=(11*sat);
assign divisor2w=20;
divide divider2(
.clk(clk2),
.dividend(dividend2w),
.divisor(divisor2w),
.quotient(quotient2w),
.remainder(remainder2w),
.rfd(rfd2w)
);
assign dividend3w=(4*hue + 6*sat);
assign divisor3w=20;
divide divider3(
.clk(clk2),
.dividend(dividend3w),
.divisor(divisor3w),
.quotient(quotient3w),
.remainder(remainder3w),
.rfd(rfd3w)
);
sqrt S2(.clk(clk2),.x_in(den1sqdw),.x_out(den1w),.rdy(rdy1w));
sqrt S3(.clk(clk2),.x_in(den2sqdw),.x_out(den2w),.rdy(rdy2w));
always@(posedge clk2)
begin
quotient1=quotient1w;
quotient2=quotient2w;
quotient3=quotient3w;
den1=den1w;den2=den2w;
RGB[7:0] = lum + quotient2;//((11*sat)/20); //red
RGB[15:8] = lum - quotient3;//((4*hue + 6*sat)/20); //green
RGB[23:16] = lum +(hue/2); //blue
tempx1=tempx1pre; tempy1=tempy1pre; tempz1=tempz1pre; tempnx1=tempnx1pre;
tempny1=tempny1pre; tempnz1=tempnz1pre;
tempx2=tempx2pre; tempy2=tempy2pre; tempz2=tempz2pre; tempnx2=tempnx2pre;
tempny2=tempny2pre; tempnz2=tempnz2pre;
temp1in=temp1inw;temp2in=temp2inw;
if(temp1in==1)begin object=9;lum=glum;end
else if(temp1in==2)
```

```
begin
object = temp1object;
if((tempx2-tempx1<2)&&(tempy2-tempy1<2)&&(tempz1-tempz2<2))
begin
den1sqd=((tempnx1*tempnx1)+(tempny1*tempny1)+(tempnz1*tempnz1));
den2sqd=(((light-tempx1)*(light-tempx1))+((light-tempy1)*(light-tempy1))+((light-
tempz1)*(lighttempz1)));
lum=quotient1;
end
else
begin
lum=glum;
end
end
if (object==1)begin hue=85;sat=137;end
if (object==2)begin hue=0;sat=80; end
if (object==3)begin hue=153;sat=195; end
if (object==4)begin hue=200;sat=240; end
if (object==5)begin hue=40;sat=240; end
if (object==6)begin hue=0;sat=240; end
if (object==7)begin hue=160;sat=0; end
if (object==8)begin hue=160;sat=0;lum=0;end
end
endmodule
```

EYE_PLANE_LINE

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 18:33:14 05/06/2007
// Design Name:
// Module Name: Eye_Plane_Line
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module Eye_Plane_Line(E_P_Lx, E_P_Ly, E_P_Lz,
E_P_Lx2, E_P_Ly2, E_P_Lz2,pixel_count,line_count,clk);
output reg signed [15:0]E_P_Lx;
output reg signed [15:0]E_P_Ly;
output reg signed [15:0]E_P_Lz;
output reg signed [15:0]E_P_Lx2;
output reg signed [15:0]E_P_Ly2;
output reg signed [15:0]E_P_Lz2;
input [10:0]pixel_count;
input [10:0]line_count; //1024 = 0b10000000000
input clk;
reg signed[15:0] pixel;
reg signed[15:0] line;
reg signed[15:0] screenx = 300;
reg signed[15:0] screeny = 1200;
reg signed[15:0] screenz = 3000;
reg signed[15:0] eyx=0;
reg signed[15:0] eyy=0;
reg signed[15:0] eyz=2500;
reg signed[15:0] psx = -60;
reg signed[15:0] psy = -80;
reg signed[15:0] psz = 0;
reg signed[15:0] lsx = 0;
reg signed[15:0] lsy = 0;
reg signed[15:0] lsz = 1000;
reg signed[15:0] pixel_total = 640;
reg signed[15:0] line_total = 480;
reg signed[15:0] quotient1;
wire [15:0]quotient1w;
wire [15:0]divisor1w;
wire [15:0]dividend1w;
wire [15:0]remainder1w;
wire rfd1w;
assign dividend1w=((psz-lsz)*line);
assign divisor1w=line_total;
divide divider4(
.clk(clk),
.dividend(dividend1w),
.divisor(divisor1w),
.quotient(quotient1w),
.remainder(remainder1w),
.rfd(rfd1w)
);
always@(posedge clk)
begin
quotient1=quotient1w;
E_P_Lx = eyx;
E_P_Ly = eyy;
E_P_Lz = eyz;
pixel = pixel_count;
line = line_count;
E_P_Lx2 = (screenx + (((psx)*pixel)/64));
E_P_Ly2 = (screeny + (((psy)*pixel)/64));
E_P_Lz2 = (screenz + quotient1);
end
endmodule
```

GET PLANES INTERSECTIONS

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 19:29:10 05/08/2007
// Design Name:
// Module Name: Get_Plane_Intersections
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
```

```
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////
module Get_Plane_Intersections(x,y,z,Pin,pNVx,pNVy,pNVz,
E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2,object,clk);
output reg [7:0]object;
input signed [15:0]E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2;
input clk;
output reg signed[15:0] x,y,z,pNVx,pNVy,pNVz;
reg signed[15:0] tpz=1000;
reg signed[15:0] BR=100;
reg signed[15:0] BR2=100; //mm
reg signed[15:0] z0 = 1000;
reg signed[15:0] y0=2000;
reg signed[15:0] x0=2000; //mm
reg signed[15:0] DP = 50; //mm
reg signed[15:0] R = 30;//radius of all balls
reg signed[15:0] xt1,yt1,zt1,pNVxt1,pNVyt1,pNVzt1;
reg signed[15:0] xt2,yt2,zt2,pNVxt2,pNVyt2,pNVzt2;
reg signed[15:0] xt3,yt3,zt3,pNVxt3,pNVyt3,pNVzt3;
reg [7:0]objectt1,objectt2,objectt3;
reg bool;
output reg [2:0]Pin;
reg [2:0]Pint1,Pint2,Pint3;
wire signed [15:0] xw,yw,zw,pNVxw,pNVyw,pNVzw;
wire signed [15:0] tpzw;
wire signed [15:0] BRw;
wire signed [15:0] BR2w;
wire signed [15:0] z0w;
wire signed [15:0] y0w;
wire signed [15:0] x0w;
wire signed [15:0] DPw;
wire signed [15:0] Rw;
wire signed [15:0] xt1w,yt1w,zt1w,pNVxt1w,pNVyt1w,pNVzt1w;
wire signed [15:0] xt2w,yt2w,zt2w,pNVxt2w,pNVyt2w,pNVzt2w;
wire signed [15:0] xt3w,yt3w,zt3w,pNVxt3w,pNVyt3w,pNVzt3w;
wire signed [7:0]objectt1w,objectt2w,objectt3w;
//wire signed boolw;
wire [2:0]Pinw;
wire signed [2:0]Pint1w,Pint2w,Pint3w;
assign tpzw=tpz;assign BRw=BR;
assign BR2w=BR2;assign z0w=z0;
assign y0w=y0;assign x0w=x0;
assign DPw=DP;assign Rw=R;
wire [15:0]zero;
wire [3:0]one;
wire [3:0]two;
wire signed [15:0]x02w;
wire signed [15:0]x02MBRw;
wire signed [15:0]x0BRw;
wire signed [15:0]y01w;
wire [3:0]three;
wire [15:0]y01MBR2w;
wire [15:0]y0BR2w;
wire [15:0]z0MDPw;
assign x02w=x0+2000;
assign x0BRw=x0+BR;
assign x02MBRw=x0+2000-BR;
assign y01w=y0+1000;
assign zero=0;
assign one=1;
assign two=2;
assign three=3;
assign y01MBR2w=y0+1000-BR2;
assign y0BR2w=y0+BR2;
assign z0MDPw=z0-DP;
Cube_Intersections
CI2(.Pxct(xt1w),.Pyct(yt1w),.Pzct(zt1w),.PNVxct(pNVxt1w),.PNVyct(pNVyt1w),.PNVzct(p
NVzt1w),
.
Pint(Pint1w),.objectt(objectt1w),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly),
.E_P_Lz(E_P_Lz), .E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.dim1(two), .dim2(one), .dim3(two),
.dim4(one), .dim5(one), .dim6(one),
.object1(two), .object2(two), .object3(two),
.object4(two), .object5(two), .object6(two),
.PL1x1(x0w),.PL1y1(y0w),.PL1z1(z0w),
.PL1x2(x02w),.PL1y2(y0w),.PL1z2(z0w),
.PL1x3(x0w),.PL1y3(y0w),.PL1z3(zero),
.PL2x1(x0w),.PL2y1(y0w),.PL2z1(z0w),
.PL2x2(x0w),.PL2y2(y01w),.PL2z2(z0w),
.PL2x3(x0w),.PL2y3(y0w),.PL2z3(zero),
.PL3x1(x0w),.PL3y1(y01w),.PL3z1(z0w),
.PL3x2(x0w),.PL3y2(y01w),.PL3z2(zero),
.PL3x3(x02w),.PL3y3(y01w),.PL3z3(z0w),
.PL4x1(x02w),.PL4y1(y0w),.PL4z1(z0w),
.PL4x2(x02w),.PL4y2(y0w),.PL4z2(zero),
.PL4x3(x02w),.PL4y3(y01w),.PL4z3(z0w),
.PL5x1(x0w),.PL5y1(y0w),.PL5z1(z0w),
.PL5x2(x0BRw),.PL5y2(y0w),.PL5z2(z0w),
.PL5x3(x0w),.PL5y3(y01w),.PL5z3(z0w),
.PL6x1(x02w),.PL6y1(y01w),.PL6z1(z0w),
.
PL6x2(x02MBRw),.PL6y2(y01w),.PL6z2(z0w),
.PL6x3(x02w),.PL6y3(y0w),.PL6z3(z0w),
.clk(clk));
Cube_Intersections
CI3(.Pxct(xt2w),.Pyct(yt2w),.Pzct(zt2w),.PNVxct(pNVxt2w),.PNVyct(pNVyt2w),.PNVzct(p
NVzt2w),
.Pint(Pint2w),.objectt(objectt2w),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly),
.E_P_Lz(E_P_Lz), .E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.dim1(three), .dim2(three), .dim3(two),
.dim4(two), .dim5(one), .dim6(one),
.object1(two), .object2(two), .object3(one),
.object4(one), .object5(one), .object6(one),
.PL1x1(x0BRw),.PL1y1(y0w),.PL1z1(z0w),
.
PL1x2(x0BRw),.PL1y2(y0BR2w),.PL1z2(z0w),
.
PL1x3(x02MBRw),.PL1y3(y0w),.PL1z3(z0w),
.
PL2x1(x0BRw),.PL2y1(y01w),.PL2z1(z0w),
.
PL2x2(x0BRw),.PL2y2(y01MBR2w),.PL2z2(z0w),
.
PL2x3(x02MBRw),.PL2y3(y01w),.PL2z3(z0w),
.
PL3x1(x0BRw),.PL3y1(y0BR2w),.PL3z1(z0w),
.
PL3x2(x0BRw),.PL3y2(y0BR2w),.PL3z2(z0MDPw),
.
PL3x3(x02MBRw),.PL3y3(y0BR2w),.PL3z3(z0w),
.
PL4x1(x0BRw),.PL4y1(y01MBR2w),.PL4z1(z0w),
.
PL4x2(x0BRw),.PL4y2(y01MBR2w),.PL4z2(z0MDPw),
.
PL4x3(x02MBRw),.PL4y3(y01MBR2w),.PL4z3(z0w),
.
PL5x1(x0BRw),.PL5y1(y0BR2w),.PL5z1(z0w),
.
PL5x2(x0BRw),.PL5y2(y0BR2w),.PL5z2(z0MDPw),
.
PL5x3(x0BRw),.PL5y3(y01MBR2w),.PL5z3(z0w),
.
PL6x1(x02MBRw),.PL6y1(y0BR2w),.PL6z1(z0w),
.
PL6x2(x02MBRw),.PL6y2(y0BR2w),.PL6z2(z0MDPw),
.
PL6x3(x02MBRw),.PL6y3(y01MBR2w),.PL6z3(z0w),
.clk(clk));
GetPlaneIntersection GPI1(.px1(x0BRw),.py1(y0BR2w),.pz1(z0w),
.px2(x02MBRw),.py2(y0+BR2),.pz2(z0w),
.px3(x0+BR),.py3(y01MBR2w),.pz3(z0w),.LDim(one),
.x(xt3w),.y(yt3w),.z(zt3w),.Pin(Pint3w),.pNVx(pNVxt3w),.pNVy(pNVyt3w),.pNVz(pNVzt3w
),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk)
);
Closest C2(.p1x(xt1),.p1y(yt1),.p1z(zt1),
.p2x(xt2),.p2y(yt2),.p2z(zt2),
.p3x(xt3),.p3y(yt3),.p3z(zt3),
.p4x(xt1),.p4y(yt1),.p4z(zt1),
.p5x(xt1),.p5y(yt1),.p5z(zt1),
.p6x(xt1),.p6y(yt1),.p6z(zt1),
.P1in(Pint1), .P2in(Pint2), .P3in(Pint3),
.P4in(zero), .P5in(zero), .P6in(zero),
.Pin(Pinw), .Pxc(xw),.Pyc(yw),.Pzc(zw),.clk(clk)
);
always@(posedge clk)
begin
if((x==xt1)&&(y==yt1)&&(z==zt1))begin
pNVx=pNVxt1;pNVy=pNVyt1;pNVz=pNVzt1;object=objectt1;end
if((x==xt2)&&(y==yt2)&&(z==zt2))begin
pNVx=pNVxt2;pNVy=pNVyt2;pNVz=pNVzt2;object=objectt2; end
if((x==xt3)&&(y==yt3)&&(z==zt3))begin
pNVx=pNVxt3;pNVy=pNVyt3;pNVz=pNVzt3;object=objectt3;end
////////////////////////////////
x=xw; y=yw; z=zw;
xt1=xt1w; yt1=yt1w; zt1=zt1w; pNVxt1=pNVxt1w; pNVyt1=pNVyt1w; pNVzt1=pNVzt1w;
xt2=xt2w; yt2=yt2w; zt2=zt2w; pNVxt2=pNVxt2w; pNVyt2=pNVyt2w; pNVzt2=pNVzt2w;
xt3=xt3w; yt3=yt3w; zt3=zt3w; pNVxt3=pNVxt3w; pNVyt3=pNVyt3w; pNVzt3=pNVzt3w;
```

```
objectt1=objectt1w; objectt2=objectt2w; objectt3=objectt3w;
Pint1=Pint1w; Pint2=Pint2w; Pint3=Pint3w;
////////////////////////////////////
Pin=Pinw;
objectt3=1;
if((Pin!=1)&&(Pin!=2))begin Pin=4;end
if(Pint3)
begin
bool=
(( (xt3-(x0+BR))*(xt3-(x0+BR)) )+((yt3-(y0+BR2))*(yt3-(y0+BR2)))<=R*R)|
(((xt3-(x0+BR))*(xt3-(x0+BR)))+((yt3-(y0+1000-BR2))*(yt3-(y0+1000-BR2)))<=R*R)|
(((xt3-(x0+2000-BR))*(xt3-(x0+2000-BR)))+((yt3-(y0+BR2))*(yt3-(y0+BR2)))<=2*R)|
(((xt3-(x0+2000-BR))*(xt3-(x0+2000-BR)))+((yt3-(y0+1000-BR2))*(yt3-(y0+1000-
BR2)))<=R*R)|
(((xt3-(x0+BR+1000))*(xt3-(x0+BR+1000)))+((yt3-(y0+BR2))*(yt3-(y0+BR2)))<=2*R)|
(((xt3-(x0+BR+1000))*(xt3-(x0+BR+1000)))+((yt3-(y0+1000-BR2))*(yt3-(y0+1000-
BR2)))<=R*R);
if(bool)objectt3=9;
end
////////////////////////////////////
end
endmodule
```

GET PLANE INTERSECTION

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 13:57:07 05/08/2007
// Design Name:
// Module Name: GetIntersection
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module GetPlaneIntersection(px1,py1,pz1,px2,py2,pz2,px3,py3,pz3,LDim,
x,y,z,Pin,pNVx,pNVy,pNVz,
E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2,clk
);
input [15:0]E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2;
input [15:0]px1,py1,pz1,px2,py2,pz2,px3,py3,pz3;
input [3:0] LDim;
input clk;
output reg signed[15:0] x,y,z,pNVx,pNVy,pNVz;
output reg [2:0]Pin;
reg signed[15:0] x1,x2,x3,y1,y2,y3,z1,z2,z3;
reg signed[15:0] A,B,C,D,Den,Num,u,xc1,xc2,xc3,xc4,yc1,yc2,yc3,yc4,v,u2,numV,denV;
///////////////////////////////////////////////
//divide stuff
reg signed[15:0] quotient1;
```

```
wire [15:0]quotient1w;
wire [15:0]divisor1w;
wire [15:0]dividend1w;
wire [15:0]remainder1w;
wire rfd1w;
reg signed[15:0] quotient2;
wire [15:0]quotient2w;
wire [15:0]divisor2w;
wire [15:0]dividend2w;
wire [15:0]remainder2w;
wire rfd2w;
reg signed[15:0] quotient3;
wire [15:0]quotient3w;
wire [15:0]divisor3w;
wire [15:0]dividend3w;
wire [15:0]remainder3w;
wire rfd3w;
reg signed[15:0] quotient4;
wire [15:0]quotient4w;
wire [15:0]divisor4w;
wire [15:0]dividend4w;
wire [15:0]remainder4w;
wire rfd4w;
//11111010000000
reg signed[15:0] quotient5;
wire [15:0]quotient5w;
wire [15:0]divisor5w;
wire [15:0]dividend5w;
wire [15:0]remainder5w;
wire rfd5w;
reg signed[15:0] quotient6;
wire [15:0]quotient6w;
wire [15:0]divisor6w;
wire [15:0]dividend6w;
wire [15:0]remainder6w;
wire rfd6w;
////////////////////////////////////////////
//quotient1 ((10**4)*(yc2-yc1)*(xc4-xc1))/(xc2-xc1)
//quotient2 100*(yc2-yc1)*(xc3-xc1)/(xc2-xc1)
//quotient3 numV/denV;
//quotient4 (100*(xc4-xc1))/(xc2-xc1);
//quotient5 (v*(xc3-xc1))/(xc2-xc1)
assign dividend1w =((10**4)*(yc2-yc1)*(xc4-xc1));
assign divisor1w=(xc2-xc1);
divide divider10(
.clk(clk),
.dividend(dividend1w),
.divisor(divisor1w),
.quotient(quotient1w),
.remainder(remainder1w),
.rfd(rfd1w)
);
assign dividend2w=100*(yc2-yc1)*(xc3-xc1);
assign divisor2w=(xc2-xc1);
divide divider5(
.clk(clk),
.dividend(dividend2w),
.divisor(divisor2w),
.quotient(quotient2w),
.remainder(remainder2w),
.rfd(rfd2w)
);
assign dividend3w=numV;
assign divisor3w=denV;
divide divider6(
.clk(clk),
.dividend(dividend3w),
.divisor(divisor3w),
.quotient(quotient3w),
.remainder(remainder3w),
.rfd(rfd3w)
);
assign dividend4w=(100*(xc4-xc1));
assign divisor4w=(xc2-xc1);
divide divider7(
.clk(clk),
.dividend(dividend4w),
.divisor(divisor4w),
.quotient(quotient4w),
.remainder(remainder4w),
.rfd(rfd4w)
);
assign dividend5w=(v*(xc3-xc1));
assign divisor5w=(xc2-xc1) ;
divide divider8(
.clk(clk),
.dividend(dividend5w),
.divisor(divisor5w),
.quotient(quotient5w),
.remainder(remainder5w),
.rfd(rfd5w)
);
assign dividend6w=Num;
assign divisor6w=Den;
divide divider9(
.clk(clk),
.dividend(dividend6w),
.divisor(divisor6w),
.quotient(quotient6w),
.remainder(remainder6w),
.rfd(rfd6w)
);
always@(posedge clk)
begin
quotient1=quotient1w;
quotient2=quotient2w;
quotient3=quotient3w;
quotient4=quotient4w;
quotient5=quotient5w;
quotient6=quotient6w;
if((Pin!=1)&&(Pin!=2))begin Pin=4;end
x1=px1;
x2=px2;
x3=px3;
y1=py1;
y2=py2;
y3=py3;
z1=pz1;
```

```
z2=pz2;
z3=pz3;
A = y1 *(z2 - z3) + y2 *(z3 - z1) + y3 *(z1 - z2);
B = z1 *(x2 - x3) + z2 *(x3 - x1) + z3* (x1 - x2);
C = x1 *(y2 - y3) + x2 *(y3 - y1) + x3 *(y1 - y2);
D = -(x1 *(y2* z3 - y3* z2) + x2 *(y3* z1 - y1 *z3) + x3* (y1* z2 - y2* z1));
pNVx=A;
pNVy=B;
pNVz=C;
Den = (A*(E_P_Lx2 - E_P_Lx)+B*(E_P_Ly2 - E_P_Ly)+C*(E_P_Lz2 - E_P_Lz));
Num = ((A * E_P_Lx + B * E_P_Ly + C * E_P_Lz + D)*1024);
if(Den==0)
begin
Pin=1;
x=0;y=0;z=0;
pNVx=0;pNVy=0;pNVz=0;
end
else
begin
u=(quotient6/1024);
x=(u*(E_P_Lx2 - E_P_Lx));
y=(u*(E_P_Ly2 - E_P_Ly));
z=(u*(E_P_Lz2 - E_P_Lz));
if(LDim==3) //all point have same z value;
begin
xc1=px1;
xc2=px2;
xc3=px3;
xc4=x;
yc1=py1;
yc2=py2;
yc3=py3;
yc4=y;
end
if(LDim==2) //all points have same y value;
begin
xc1=px1;
xc2=px2;
xc3=px3;
xc4=x;
yc1=pz1;
yc2=pz2;
yc3=pz3;
yc4=z;
end
if(LDim==1) //all points have same x value;
begin
xc1=pz1;
xc2=pz2;
xc3=pz3;
xc4=z;
yc1=py1;
yc2=py2;
yc3=py3;
yc4=y;
end
numV =(((10000)*(yc4-yc1)) - quotient1);
denV= ((100*(yc3-yc1)) - quotient2);
v = quotient3;
```

```
u2 = (quotient4 - quotient5);
if((u2<=100)&&(v<=100)&&(u2>=0)&&(v>=0))begin Pin=2;end
else if((u2>100)|(v>100)|(u2<0)|(v<0))begin Pin=1;end
end
end
endmodule
```

CLOSEST

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 19:15:16 05/09/2007
// Design Name:
// Module Name: Closest
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////
module Closest(p1x,p1y,p1z,p2x,p2y,p2z,p3x,p3y,p3z,
p4x,p4y,p4z,p5x,p5y,p5z,p6x,p6y,p6z,
P1in, P2in, P3in, P4in, P5in, P6in, Pin, Pxc,Pyc,Pzc,clk
);
input clk;
input [2:0]P1in, P2in, P3in, P4in, P5in, P6in;
input signed [15:0]p1x,p1y,p1z;
input signed [15:0]p2x,p2y,p2z;
input signed [15:0]p3x,p3y,p3z;
input signed [15:0]p4x,p4y,p4z;
input signed [15:0]p5x,p5y,p5z;
input signed [15:0]p6x,p6y,p6z;
output reg signed[15:0] Pxc,Pyc,Pzc;
output reg [2:0]Pin;
reg signed[15:0] point1;
reg signed[15:0] point2;
reg signed[15:0] point3;
reg signed[15:0] point4;
reg signed[15:0] point5;
reg signed[15:0] point6;
reg signed[15:0] pointc1;
reg signed[15:0] pointc2;
reg signed[15:0] pointc3;
reg signed[15:0] pointc4;
reg signed[15:0] pointf;
reg [2:0]pointc1in;
reg [2:0]pointc2in;
reg [2:0]pointc3in;
reg [2:0]pointc4in;
reg [2:0]pointfin;
always@(posedge clk)
```

```
begin
if((Pin!=1)&&(Pin!=2))begin Pin=4;end
///////////Get closest point
point1 = (p1x*p1x +p1y*p1y +p1z*p1z);
point2 = (p2x*p2x +p2y*p2y +p2z*p2z);
point3 = (p3x*p3x +p3y*p3y +p3z*p3z);
point4 = (p4x*p4x +p4y*p4y +p4z*p4z);
point5 = (p5x*p5x +p5y*p5y +p5z*p5z);
point6 = (p6x*p6x +p6y*p6y +p6z*p6z);
if((P1in==1)&&(P2in==1))begin pointc1in=1;end
else if((P1in==1)&&(P2in==2))begin pointc1in=P2in;pointc1=point2;end
else if((P2in==1)&&(P1in==2))begin pointc1in=P1in;pointc1=point1;end
else if((P2in==2)&&(P1in==2)&&(point1>=point2))begin pointc1in=P2in;
pointc1=point2;end
else if((P2in==2)&&(P1in==2))begin pointc1in=P1in; pointc1=point1;end
if((P3in==1)&&(P4in==1))begin pointc2in=1;end
else if((P3in==1)&&(P4in==2))begin pointc2in=P4in;pointc2=point4;end
else if((P4in==1)&&(P3in==2))begin pointc2in=P3in;pointc2=point3;end
else if((P4in==2)&&(P3in==2)&&(point3>=point4))begin pointc2in=P4in;
pointc2=point4;end
else if((P4in==2)&&(P3in==2))begin pointc2in=P3in; pointc2=point3;end
if((P5in==1)&&(P6in==1))begin pointc3in=1;end
else if((P5in==1)&&(P6in==2))begin pointc3in=P6in;pointc3=point6;end
else if((P5in==2)&&(P6in==1))begin pointc3in=P5in;pointc3=point5;end
else if((P5in==2)&&(P6in==2)&&(point5>=point6))begin pointc3in=P6in;
pointc3=point6;end
else if((P5in==2)&&(P6in==2))begin pointc3in=P5in; pointc3=point5;end
if((pointc1in==1)&&(pointc2in==1))begin pointc4in=1;end
else if((pointc1in==1)&&(pointc2in==2))begin
pointc4in=pointc2in;pointc4=pointc2;end
else if((pointc1in==2)&&(pointc2in==1))begin
pointc4in=pointc1in;pointc4=pointc1;end
else if((pointc1in==2)&&(pointc2in==2)&&(pointc1>=pointc2))begin
pointc4in=pointc2in;
pointc4=pointc2;end
else if((pointc1in==2)&&(pointc2in==2))begin pointc4in=pointc1in;
pointc4=pointc1;end
if((pointc3in==1)&&(pointc4in==1))begin pointfin=1;end
else if((pointc3in==1)&&(pointc4in==2))begin pointfin=pointc4in;pointf=pointc4;end
else if((pointc3in==2)&&(pointc4in==1))begin pointfin=pointc3in;pointf=pointc3;end
else if((pointc3in==2)&&(pointc4in==2)&&(pointc3>=pointc4))begin
pointfin=pointc4in;
pointf=pointc4;end
else if((pointc3in==2)&&(pointc4in==2))begin pointfin=pointc3in; pointf=pointc3;end
if(pointfin==1)Pin=1;
else
begin
case(pointf)
point1:
begin
Pxc=p1x;Pyc=p1y;Pzc=p1z;
Pin=P1in;
//object=object1;
end
point2:
begin
Pxc=p2x;Pyc=p2y;Pzc=p2z;
Pin=P2in;
//object=object2;
```

```
end
point3:
begin
Pxc=p3x;Pyc=p3y;Pzc=p3z;
Pin=P3in;
//object=object3;
end
point4:
begin
Pxc=p4x;Pyc=p4y;Pzc=p4z;
Pin=P4in;
//object=object4;
end
point5:
begin
Pxc=p5x;Pyc=p5y;Pzc=p5z;
Pin=P5in;
//object=object5;
end
point6:
begin
Pxc=p6x;Pyc=p6y;Pzc=p6z;
Pin=P6in;
//object=object6;
end
endcase
end//else
end//always
endmodule
```

GET BALL INTERSECTIONS

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 11:30:38 05/10/2007
// Design Name:
// Module Name: Get_Ball_Intersections
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module Get_Ball_Intersections(bin,px,py,pz,pxnv,pynv,pznv,object,
bx1,bx2,bx3,bx4,bx5,bx6,bx7,bx8,bx9,bx10,bx11,bx12,bx13,bx14,bx15,
by1,by2,by3,by4,by5,by6,by7,by8,by9,by10,by11,by12,by13,by14,by15,
bp1,bp2,bp3,bp4,bp5,bp6,bp7,bp8,bp9,bp10,bp11,bp12,bp13,bp14,bp15,
E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2,
clk
);
output reg [2:0]bin;
output reg signed [15:0] px,py,pz,pxnv,pynv,pznv;
input signed [15:0]
bx1,bx2,bx3,bx4,bx5,bx6,bx7,bx8,bx9,bx10,bx11,bx12,bx13,bx14,bx15;
input signed [15:0]
by1,by2,by3,by4,by5,by6,by7,by8,by9,by10,by11,by12,by13,by14,by15;
input bp1,bp2,bp3,bp4,bp5,bp6,bp7,bp8,bp9,bp10,bp11,bp12,bp13,bp14,bp15;
input signed [15:0] E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2;
input clk;
reg signed[15:0] bzw=1030;
wire signed[15:0]bz;
assign bz=bzw;
wire [2:0]bin1;
wire signed [15:0] px1,py1,pz1,pxnv1,pynv1,pznv1;
wire [2:0]bin2;
wire signed [15:0] px2,py2,pz2,pxnv2,pynv2,pznv2;
wire [2:0]bin3;
wire signed [15:0] px3,py3,pz3,pxnv3,pynv3,pznv3;
wire [2:0]bin4;
wire signed [15:0] px4,py4,pz4,pxnv4,pynv4,pznv4;
wire [2:0]bin5;
wire signed [15:0] px5,py5,pz5,pxnv5,pynv5,pznv5;
wire [2:0]bin6;
wire signed [15:0] px6,py6,pz6,pxnv6,pynv6,pznv6;
wire [2:0]bin7;
wire signed [15:0] px7,py7,pz7,pxnv7,pynv7,pznv7;
wire [2:0]bin8;
wire signed [15:0] px8,py8,pz8,pxnv8,pynv8,pznv8;
wire [2:0]bin9;
wire signed [15:0] px9,py9,pz9,pxnv9,pynv9,pznv9;
wire [2:0]bin10;
wire signed [15:0] px10,py10,pz10,pxnv10,pynv10,pznv10;
wire [2:0]bin11;
wire signed [15:0] px11,py11,pz11,pxnv11,pynv11,pznv11;
wire [2:0]bin12;
wire signed [15:0] px12,py12,pz12,pxnv12,pynv12,pznv12;
wire [2:0]bin13;
wire signed [15:0] px13,py13,pz13,pxnv13,pynv13,pznv13;
wire [2:0]bin14;
wire signed [15:0] px14,py14,pz14,pxnv14,pynv14,pznv14;
wire [2:0]bin15;
wire signed [15:0] px15,py15,pz15,pxnv15,pynv15,pznv15;
wire signed [15:0] tempx1,tempy1,tempz1;
wire signed [15:0] tempx2,tempy2,tempz2;
wire signed [15:0] tempx3,tempy3,tempz3;
wire [2:0]temp1in,temp2in,temp3in;
reg signed[15:0] tempx1i,tempy1i,tempz1i;
reg signed[15:0] tempx2i,tempy2i,tempz2i;
reg signed[15:0] tempx3i,tempy3i,tempz3i;
reg [2:0]temp1ini,temp2ini,temp3ini;
wire signed [15:0] zero;
assign zero=0;
reg [7:0]yellowball=5;
reg [7:0]redball=6;
reg [7:0]whiteball=7;
reg [7:0]blackball=8;
output reg [7:0]object;
//pxt,pyt,pzt,pxnvt,pynvt,pznvt,bpt
Ball_Intersection B1(.bx(bx1),.by(by1),
.bin(bin1),
.pxt(px1),.pyt(py1),.pzt(pz1),.pxnvt(pxnv1),.pynvt(pynv1),.pznvt(pznv1),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp1)
);
Ball_Intersection B2(.bx(bx2),.by(by2),
.bin(bin2),
.pxt(px2),.pyt(py2),.pzt(pz2),.pxnvt(pxnv2),.pynvt(pynv2),.pznvt(pznv2),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp2)
);
Ball_Intersection B3(.bx(bx3),.by(by3),
.bin(bin3),
.pxt(px3),.pyt(py3),.pzt(pz3),.pxnvt(pxnv3),.pynvt(pynv3),.pznvt(pznv3),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp3)
);
Ball_Intersection B4(.bx(bx4),.by(by4),
.bin(bin4),
.pxt(px4),.pyt(py4),.pzt(pz4),.pxnvt(pxnv4),.pynvt(pynv4),.pznvt(pznv4),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp4)
);
Ball_Intersection B5(.bx(bx5),.by(by5),
.bin(bin5),
.pxt(px5),.pyt(py5),.pzt(pz5),.pxnvt(pxnv5),.pynvt(pynv5),.pznvt(pznv5),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp5)
);
Ball_Intersection B6(.bx(bx6),.by(by6),
.bin(bin6),
.pxt(px6),.pyt(py6),.pzt(pz6),.pxnvt(pxnv6),.pynvt(pynv6),.pznvt(pznv6),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp6)
);
Ball_Intersection B7(.bx(bx7),.by(by7),
.bin(bin7),
.pxt(px7),.pyt(py7),.pzt(pz7),.pxnvt(pxnv7),.pynvt(pynv7),.pznvt(pznv7),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp7)
);
Ball_Intersection B8(.bx(bx8),.by(by8),
.bin(bin8),
.pxt(px8),.pyt(py8),.pzt(pz8),.pxnvt(pxnv8),.pynvt(pynv8),.pznvt(pznv8),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp8)
);
Ball_Intersection B9(.bx(bx9),.by(by9),
.bin(bin9),
.pxt(px9),.pyt(py9),.pzt(pz9),.pxnvt(pxnv9),.pynvt(pynv9),.pznvt(pznv9),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp9)
);
Ball_Intersection B10(.bx(bx10),.by(by10),
.bin(bin10),
.pxt(px10),.pyt(py10),.pzt(pz10),.pxnvt(pxnv10),.pynvt(pynv10),.pznvt(pznv10),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp10)
);
Ball_Intersection B11(.bx(bx11),.by(by11),
.bin(bin11),
.pxt(px11),.pyt(py11),.pzt(pz11),.pxnvt(pxnv11),.pynvt(pynv11),.pznvt(pznv11),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp11)
);
Ball_Intersection B12(.bx(bx12),.by(by12),
.bin(bin12),
.pxt(px12),.pyt(py12),.pzt(pz12),.pxnvt(pxnv12),.pynvt(pynv12),.pznvt(pznv12),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp12)
);
Ball_Intersection B13(.bx(bx13),.by(by13),
.bin(bin13),
.pxt(px13),.pyt(py13),.pzt(pz13),.pxnvt(pxnv13),.pynvt(pynv13),.pznvt(pznv13),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp13)
);
Ball_Intersection B14(.bx(bx14),.by(by14),
.bin(bin14),
.pxt(px14),.pyt(py14),.pzt(pz14),.pxnvt(pxnv14),.pynvt(pynv14),.pznvt(pznv14),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp14)
);
Ball_Intersection B15(.bx(bx15),.by(by15),
.bin(bin15),
.pxt(px15),.pyt(py15),.pzt(pz15),.pxnvt(pxnv15),.pynvt(pynv15),.pznvt(pznv15),
.E_P_Lx(E_P_Lx), .E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),.clk(clk),.bpt(bp15)
);
Closest
C2(.p1x(px1),.p1y(py1),.p1z(pz1),.p2x(px2),.p2y(py2),.p2z(pz2),.p3x(px3),.p3y(py3),
.p3z(pz3),
.
p4x(px4),.p4y(py4),.p4z(pz4),.p5x(px5),.p5y(py5),.p5z(pz5),.p6x(px6),.p6y(py6),.p6z
(pz6),
.P1in(bp1), .P2in(bp2),
.P3in(bp3), .P4in(bp4), .P5in(bp5), .P6in(bp6),
.Pin(temp1in),
.Pxc(tempx1),.Pyc(tempy1),.Pzc(tempz1),.clk(clk)
);
Closest
C3(.p1x(px7),.p1y(py7),.p1z(pz7),.p2x(px8),.p2y(py8),.p2z(pz8),.p3x(px9),.p3y(py9),
.p3z(pz9),
.
p4x(px10),.p4y(py10),.p4z(pz10),.p5x(px11),.p5y(py11),.p5z(pz11),.p6x(px12),.p6y(py
12),.p6z(pz12),
.P1in(bp7), .P2in(bp8),
.P3in(bp9), .P4in(bp10), .P5in(bp11), .P6in(bp12),
.Pin(temp2in),
.Pxc(tempx2),.Pyc(tempy2),.Pzc(tempz2),.clk(clk)
);
Closest
C4(.p1x(px13),.p1y(py13),.p1z(pz13),.p2x(px14),.p2y(py14),.p2z(pz14),.p3x(px15),.p3
y(py15),.p3z(p
z15),
.
p4x(tempx1),.p4y(tempy1),.p4z(tempz1),.p5x(tempx2),.p5y(tempy2),.p5z(tempz2),.p6x(z
ero),.p6y(zer
o),.p6z(zero),
.P1in(bp13), .P2in(bp14),
.P3in(bp15), .P4in(temp1in), .P5in(temp2in), .P6in(0),
.Pin(temp3in),
.Pxc(tempx3),.Pyc(tempy3),.Pzc(tempz3),.clk(clk)
);
always@(posedge clk)
begin
if((bin!=1)&&(bin!=2))begin bin=4;end
if (temp3in==2)
begin
bin=2;
px=tempx3;py=tempy3;pz=tempz3;
if ((tempx3==bx1)&&(tempy3==by1)&&(tempz3==bzw)) pxnv=(px-bx1);pynv=(py-
by1);pznv=(pzbz);
object=yellowball;
if ((tempx3==bx2)&&(tempy3==by2)&&(tempz3==bzw)) pxnv=(px-bx2);pynv=(py-
by2);pznv=(pzbz);
object=yellowball;
if ((tempx3==bx3)&&(tempy3==by3)&&(tempz3==bzw)) pxnv=(px-bx3);pynv=(py-
by3);pznv=(pzbz);
object=yellowball;
if ((tempx3==bx4)&&(tempy3==by4)&&(tempz3==bzw)) pxnv=(px-bx4);pynv=(py-
by4);pznv=(pzbz);
object=yellowball;
if ((tempx3==bx5)&&(tempy3==by5)&&(tempz3==bzw)) pxnv=(px-bx5);pynv=(py-
by5);pznv=(pzbz);
object=yellowball;
if ((tempx3==bx6)&&(tempy3==by6)&&(tempz3==bzw)) pxnv=(px-bx6);pynv=(py-
by6);pznv=(pzbz);
object=yellowball;
if ((tempx3==bx7)&&(tempy3==by7)&&(tempz3==bzw)) pxnv=(px-bx7);pynv=(py-
by7);pznv=(pzbz);
object=yellowball;
if ((tempx3==bx8)&&(tempy3==by8)&&(tempz3==bzw)) pxnv=(px-bx8);pynv=(py-
by8);pznv=(pzbz);
object=blackball;
if ((tempx3==bx9)&&(tempy3==by9)&&(tempz3==bzw)) pxnv=(px-bx9);pynv=(py-
by9);pznv=(pzbz);
object=redball;
if ((tempx3==bx10)&&(tempy3==by10)&&(tempz3==bzw)) pxnv=(px-bx10);pynv=(pyby10);
pznv=(pz-bz);object=redball;
if ((tempx3==bx11)&&(tempy3==by11)&&(tempz3==bzw)) pxnv=(px-bx11);pynv=(pyby11);
pznv=(pz-bz);object=redball;
if ((tempx3==bx12)&&(tempy3==by12)&&(tempz3==bzw)) pxnv=(px-bx12);pynv=(pyby12);
pznv=(pz-bz);object=redball;
if ((tempx3==bx13)&&(tempy3==by13)&&(tempz3==bzw)) pxnv=(px-bx13);pynv=(pyby13);
pznv=(pz-bz);object=redball;
if ((tempx3==bx14)&&(tempy3==by14)&&(tempz3==bzw)) pxnv=(px-bx14);pynv=(pyby14);
pznv=(pz-bz);object=redball;
if ((tempx3==bx15)&&(tempy3==by15)&&(tempz3==bzw)) pxnv=(px-bx15);pynv=(pyby15);
pznv=(pz-bz);object=whiteball;
end
else if(temp3in==1)
begin
bin=1;
end
///////////////////////////////
tempx1i=tempx1 ;tempy1i= tempy1;tempz1i=tempz1;
tempx2i= tempx2;tempy2i= tempy2;tempz2i=tempz2;
```

```
tempx3i=tempx3 ;tempy3i= tempy3;tempz3i=tempz3;
temp1ini=temp1in ;temp2ini=temp2in ;temp3ini=temp3in;
//////////////////////////////
end
endmodule
```

GET BALL INTERSECTION

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 21:32:10 05/09/2007
// Design Name:
// Module Name: Ball_Intersection
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module Ball_Intersection(bx,by,E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2,
E_P_Lz2,clk,
bin, pxt,pyt,pzt,pxnvt,pynvt,pznvt,bpt
);
input signed [15:0]bx,by,E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2;
input clk,bpt;
reg bp;
output reg [2:0]bin;
output reg signed [15:0]pxt,pyt,pzt,pxnvt,pynvt,pznvt;
reg signed[15:0] px,py,pz,pxnv,pynv,pznv;
////////////////////////////////////
//divider stuff
reg signed[15:0] quotient1;
wire [15:0]quotient1w;
wire [15:0]divisor1w;
wire [15:0]dividend1w;
wire [15:0]remainder1w;
wire rfd1w;
/////////////////////////////////
reg signed[15:0] bz=1030;//tpz+R;
reg signed[15:0] x1,x2,y1,y2,z1,z2;
reg signed[15:0] R=30;
reg signed[15:0] tpz=1000;
wire[15:0]sqtempw,sqholdw;
reg signed[15:0] sqtemp;
reg signed[15:0] sqhold,sqval;
reg signed[15:0] a,b,c,den,num,u;
wire rdy;
sqrt S1(.clk(clk),.x_in(sqtempw),.x_out(sqholdw),.rdy(rdy));
assign sqtempw=sqtemp;
assign dividend1w=(0-b-sqval);
assign divisor1w=(2*a);
```

```
divide divider11(
.clk(clk),
.dividend(dividend1w),
.divisor(divisor1w),
.quotient(quotient1w),
.remainder(remainder1w),
.rfd(rfd1w)
);
always@(posedge clk)
begin
quotient1=quotient1w;
bp=bpt;
sqhold=sqholdw;
pxt=px;pyt=py;pzt=pz;pxnvt=pxnv;pynvt=pynv;pznvt=pznv;
if(bp==0)begin bin=1;end
else if((bin!=1)&&(bin!=2))begin bin=4;end
x1 = E_P_Lx; x2 = E_P_Lx2;
y1 = E_P_Ly; y2 = E_P_Ly2;
z1 = E_P_Lz; z2 = E_P_Lz2;
a = ((x2 - x1)*(x2 - x1)) + ((y2 - y1)*(y2 - y1)) + ((z2 - z1)*(z2 - z1));
b = 2*( (x2 - x1)*(x1 - bx) + (y2 - y1)*(y1 - by) +
(z2 - z1)*(z1 - bz) );
c = (bx*bx) + (by*by) + (bz*bz) + (x1*x1) + (y1*y1) + (z1*z1) -
2*(bx*x1 + by*y1 + bz*z1) - (R*R);
sqtemp=b*b-4*a*c;
if((sqtemp<0)&&(bin==4))
begin
bin=1;
end
else if((sqtemp>=0)&&(bin==4)&& rdy)
begin
bin=2;
if(rdy)
begin
sqval=sqhold;
num=(0-b-sqval);
den=(2*a);
u=quotient1;
px=u*(x2 - x1);
py=u*(y2 - y1);
pz=u*(z2 - z1);
pxnv=(px-bx);
pynv=(py-by);
pznv=(pz-bz);
end
end
end
endmodule
```

CUBE INTERSECTIONS

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 15:24:40 05/09/2007
// Design Name:
// Module Name: Cube_Intersections
// Project Name:
```

```
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module Cube_Intersections(Pxct,Pyct,Pzct,PNVxct,PNVyct,PNVzct,Pint,objectt,clk,
E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2,
E_P_Ly2, E_P_Lz2,
dim1, dim2, dim3, dim4, dim5, dim6,
object1, object2, object3, object4, object5,
object6,
PL1x1,PL1y1,PL1z1,
PL1x2,PL1y2,PL1z2,
PL1x3,PL1y3,PL1z3,
PL2x1,PL2y1,PL2z1,
PL2x2,PL2y2,PL2z2,
PL2x3,PL2y3,PL2z3,
PL3x1,PL3y1,PL3z1,
PL3x2,PL3y2,PL3z2,
PL3x3,PL3y3,PL3z3,
PL4x1,PL4y1,PL4z1,
PL4x2,PL4y2,PL4z2,
PL4x3,PL4y3,PL4z3,
PL5x1,PL5y1,PL5z1,
PL5x2,PL5y2,PL5z2,
PL5x3,PL5y3,PL5z3,
PL6x1,PL6y1,PL6z1,
PL6x2,PL6y2,PL6z2,
PL6x3,PL6y3,PL6z3
);
output reg signed [15:0]Pxct,Pyct,Pzct,PNVxct,PNVyct,PNVzct;
output reg [2:0]Pint;
output reg [7:0]objectt;
wire signed [15:0]Pxc,Pyc,Pzc,PNVxc,PNVyc,PNVzc;
wire [2:0]Pin;
wire [7:0]object;
input clk;
input [3:0]dim1;
input [3:0]dim2;
input [3:0]dim3;
input [3:0]dim4;
input [3:0]dim5;
input [3:0]dim6;
input [7:0] object1;
input [7:0] object2;
input [7:0] object3;
input [7:0] object4;
input [7:0] object5;
input [7:0] object6;
input signed [15:0]E_P_Lx, E_P_Ly, E_P_Lz, E_P_Lx2, E_P_Ly2, E_P_Lz2;
input signed [15:0] PL1x1,PL1y1,PL1z1;
input signed [15:0] PL1x2,PL1y2,PL1z2;
input signed [15:0] PL1x3,PL1y3,PL1z3;
input signed [15:0] PL2x1,PL2y1,PL2z1;
input signed [15:0] PL2x2,PL2y2,PL2z2;
input signed [15:0] PL2x3,PL2y3,PL2z3;
input signed [15:0] PL3x1,PL3y1,PL3z1;
input signed [15:0] PL3x2,PL3y2,PL3z2;
input signed [15:0] PL3x3,PL3y3,PL3z3;
input signed [15:0] PL4x1,PL4y1,PL4z1;
input signed [15:0] PL4x2,PL4y2,PL4z2;
input signed [15:0] PL4x3,PL4y3,PL4z3;
input signed [15:0] PL5x1,PL5y1,PL5z1;
input signed [15:0] PL5x2,PL5y2,PL5z2;
input signed [15:0] PL5x3,PL5y3,PL5z3;
input signed [15:0] PL6x1,PL6y1,PL6z1;
input signed [15:0] PL6x2,PL6y2,PL6z2;
input signed [15:0] PL6x3,PL6y3,PL6z3;
wire signed [15:0]p1x,p1y,p1z;
wire signed [15:0]p2x,p2y,p2z;
wire signed [15:0]p3x,p3y,p3z;
wire signed [15:0]p4x,p4y,p4z;
wire signed [15:0]p5x,p5y,p5z;
wire signed [15:0]p6x,p6y,p6z;
wire signed [15:0]p1NVx,p1NVy,p1NVz;
wire signed [15:0]p2NVx,p2NVy,p2NVz;
wire signed [15:0]p3NVx,p3NVy,p3NVz;
wire signed [15:0]p4NVx,p4NVy,p4NVz;
wire signed [15:0]p5NVx,p5NVy,p5NVz;
wire signed [15:0]p6NVx,p6NVy,p6NVz;
wire [2:0]P1in, P2in, P3in, P4in, P5in, P6in;
reg signed[15:0] p1xt,p1yt,p1zt;
reg signed[15:0] p2xt,p2yt,p2zt;
reg signed[15:0] p3xt,p3yt,p3zt;
reg signed[15:0] p4xt,p4yt,p4zt;
reg signed[15:0] p5xt,p5yt,p5zt;
reg signed[15:0] p6xt,p6yt,p6zt;
reg signed[15:0] p1NVxt,p1NVyt,p1NVzt;
reg signed[15:0] p2NVxt,p2NVyt,p2NVzt;
reg signed[15:0] p3NVxt,p3NVyt,p3NVzt;
reg signed[15:0] p4NVxt,p4NVyt,p4NVzt;
reg signed[15:0] p5NVxt,p5NVyt,p5NVzt;
reg signed[15:0] p6NVxt,p6NVyt,p6NVzt;
reg [2:0]P1int, P2int, P3int, P4int, P5int, P6int;
///////////Plane 1
GetPlaneIntersection GPI1(.px1(PL1x1),.py1(PL1y1),.pz1(PL1z1),
.
px2(PL1x2),.py2(PL1y2),.pz2(PL1z2),
.
px3(PL1x3),.py3(PL1y3),.pz3(PL1z3),
.
x(p1x),.y(p1y),.z(p1z),.Pin(P1in),
.
pNVx(p1NVx),.pNVy(p1NVy), .pNVz(p1NVz),
.LDim(dim1),
.E_P_Lx(E_P_Lx),
.E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.clk(clk));
///////////Plane 2
GetPlaneIntersection GPI2(.px1(PL2x1),.py1(PL2y1),.pz1(PL2z1),
.
px2(PL2x2),.py2(PL2y2),.pz2(PL2z2),
.
px3(PL2x3),.py3(PL2y3),.pz3(PL2z3),
.
x(p2x),.y(p2y),.z(p2z),.Pin(P2in),
.
pNVx(p2NVx),.pNVy(p2NVy), .pNVz(p2NVz),
.LDim(dim2),
.E_P_Lx(E_P_Lx),
.E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.clk(clk));
///////////Plane 3
GetPlaneIntersection GPI3(.px1(PL3x1),.py1(PL3y1),.pz1(PL3z1),
.
px2(PL3x2),.py2(PL3y2),.pz2(PL3z2),
.
px3(PL3x3),.py3(PL3y3),.pz3(PL3z3),
.
x(p3x),.y(p3y),.z(p3z),.Pin(P3in),
.
pNVx(p3NVx),.pNVy(p3NVy), .pNVz(p3NVz),
.LDim(dim3),
.E_P_Lx(E_P_Lx),
.E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.clk(clk));
///////////Plane 4
GetPlaneIntersection GPI4(.px1(PL4x1),.py1(PL4y1),.pz1(PL4z1),
.
px2(PL4x2),.py2(PL4y2),.pz2(PL4z2),
.
px3(PL4x3),.py3(PL4y3),.pz3(PL4z3),
.
x(p4x),.y(p4y),.z(p4z),.Pin(P4in),
.
pNVx(p4NVx),.pNVy(p4NVy), .pNVz(p4NVz),
.LDim(dim4),
.E_P_Lx(E_P_Lx),
.E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.clk(clk));
///////////Plane 5
GetPlaneIntersection GPI5(.px1(PL5x1),.py1(PL5y1),.pz1(PL5z1),
.
px2(PL5x2),.py2(PL5y2),.pz2(PL5z2),
.
px3(PL5x3),.py3(PL5y3),.pz3(PL5z3),
.
x(p5x),.y(p5y),.z(p5z),.Pin(P5in),
.
pNVx(p5NVx),.pNVy(p5NVy), .pNVz(p5NVz),
.LDim(dim5),
.E_P_Lx(E_P_Lx),
.E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.clk(clk));
///////////Plane 6
GetPlaneIntersection GPI6(.px1(PL6x1),.py1(PL6y1),.pz1(PL6z1),
.
px2(PL6x2),.py2(PL6y2),.pz2(PL6z2),
.
px3(PL6x3),.py3(PL6y3),.pz3(PL6z3),
.
x(p6x),.y(p6y),.z(p6z),.Pin(P6in),
.
pNVx(p6NVx),.pNVy(p6NVy), .pNVz(p6NVz),
.LDim(dim6),
.E_P_Lx(E_P_Lx),
.E_P_Ly(E_P_Ly), .E_P_Lz(E_P_Lz),
.
E_P_Lx2(E_P_Lx2), .E_P_Ly2(E_P_Ly2), .E_P_Lz2(E_P_Lz2),
.clk(clk));
Closest C1(.p1x(p1x),.p1y(p1y),.p1z(p1z),
.p2x(p2x),.p2y(p2y),.p2z(p2z),
.p3x(p3x),.p3y(p3y),.p3z(p3z),
.p4x(p4x),.p4y(p4y),.p4z(p4z),
.p5x(p5x),.p5y(p5y),.p5z(p5z),
.p6x(p6x),.p6y(p6y),.p6z(p6z),
.P1in(P1in), .P2in(P2in), .P3in(P3in),
.P4in(P4in), .P5in(P5in), .P6in(P6in),
.Pin(Pin), .Pxc(Pxc),.Pyc(Pyc),.Pzc(Pzc),.clk(clk)
);
always@(posedge clk)
begin
Pxct=Pxc;Pyct=Pyc;Pzct=Pzc;
p1xt=p1x ;p1yt=p1y ;p1zt=p1z ;
p2xt=p2x ;p2yt=p2y ;p2zt=p2z ;
p3xt=p3x ;p3yt=p3y ;p3zt=p3z ;
p4xt=p4x ;p4yt=p4y ;p4zt=p4z ;
p5xt=p5x ;p5yt=p5y ;p5zt=p5z ;
p6xt=p6x ;p6yt=p6y ;p6zt=p6z ;
p1NVxt=p1NVx;p1NVyt=p1NVy;p1NVzt=p1NVz;
p2NVxt=p2NVx;p2NVyt=p2NVy;p2NVzt=p2NVz;
p3NVxt=p3NVx;p3NVyt=p3NVy;p3NVzt=p3NVz;
p4NVxt=p4NVx;p4NVyt=p4NVy;p4NVzt=p4NVz;
p5NVxt=p5NVx;p5NVyt=p5NVy;p5NVzt=p5NVz;
p6NVxt=p6NVx;p6NVyt=p6NVy;p6NVzt=p6NVz;
P1int=P1in; P2int=P2in ; P3int=P3in ; P4int=P4in ; P5int=P5in ; P6int=P6in ;
if((Pint!=1)&&(Pint!=2))begin Pint=4;end
if(Pint==2)
begin
if((Pxct==p1x)&&(Pyct==p1y)&&(Pzct==p1z))
begin
PNVxct=p1NVx;PNVyct=p1NVy;PNVzct=p1NVz;
objectt=object1;
end
if((Pxct==p2x)&&(Pyct==p2y)&&(Pzct==p2z))
begin
PNVxct=p2NVx;PNVyct=p2NVy;PNVzct=p2NVz;
objectt=object2;
end
if((Pxct==p3x)&&(Pyct==p3y)&&(Pzct==p3z))
```

```
begin
PNVxct=p3NVx;PNVyct=p3NVy;PNVzct=p3NVz;
objectt=object3;
end
if((Pxct==p4x)&&(Pyct==p4y)&&(Pzct==p4z))
begin
PNVxct=p4NVx;PNVyct=p4NVy;PNVzct=p4NVz;
objectt=object4;
end
if((Pxct==p5x)&&(Pyct==p5y)&&(Pzct==p5z))
begin
PNVxct=p5NVx;PNVyct=p5NVy;PNVzct=p5NVz;
objectt=object5;
end
if((Pxct==p6x)&&(Pyct==p6y)&&(Pzct==p6z))
begin
PNVxct=p6NVx;PNVyct=p6NVy;PNVzct=p6NVz;
objectt=object6;
end
end
end
endmodule
```