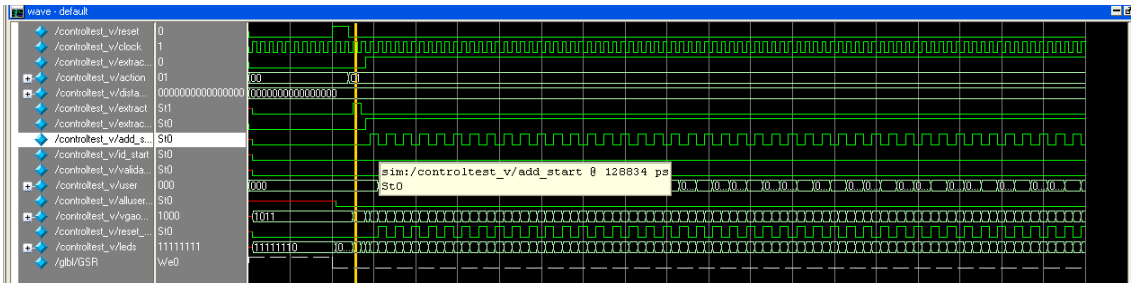
5.1 Appendix : Simulations

Control Unit:
Code:

```
reset = 1'b1;
#20
reset = 0;
action = 2'b01;
#20
extracted = 1'b1;
```

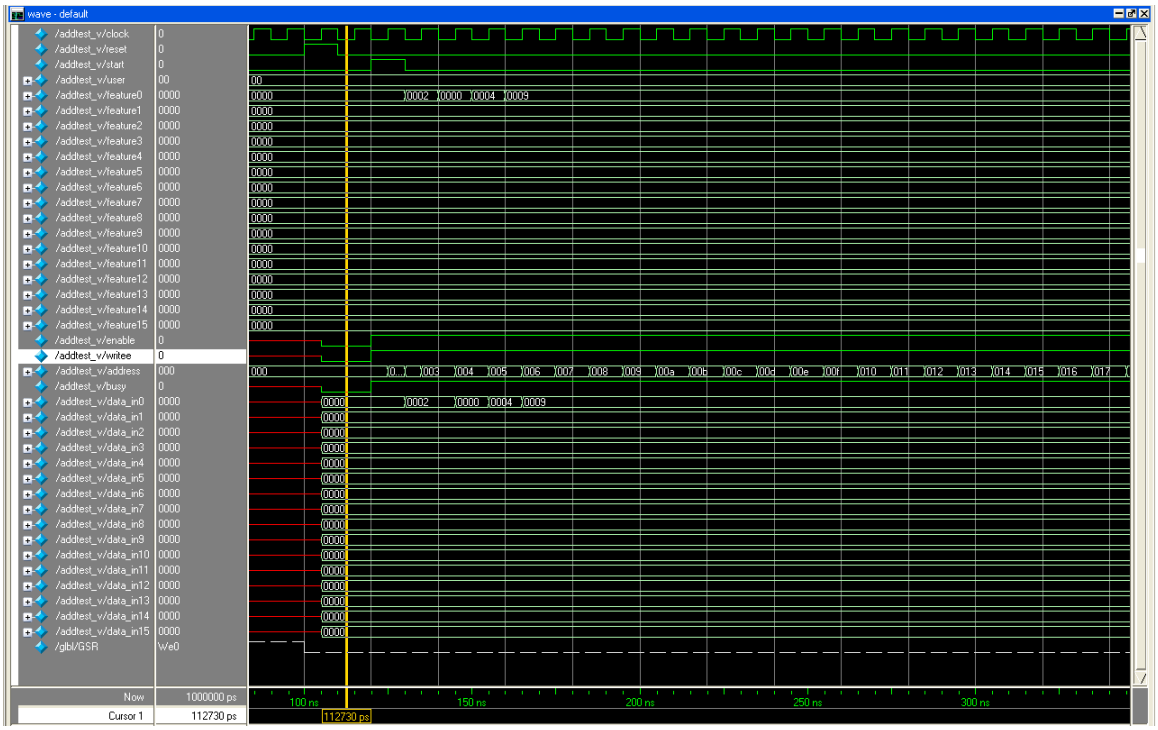
Simulation:



MEM_ ADD:
Code:

```
reset = 1'b1;
#10
reset = 0;
#10;
start = 1'b1;
#10
begin
start = 1'b0;
feature0 = 16'd2;
end
#10
feature0 = 0;
#10
feature0 = 16'd4;
#10
feature0 = 16'd9;
```

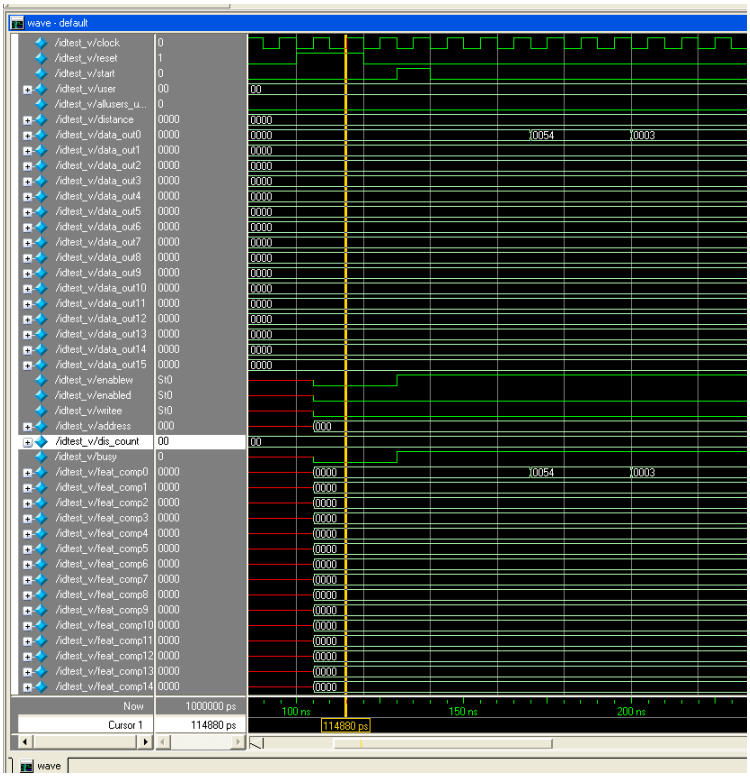
Simulation:



MEM_ID :
Code:

```
reset = 1'b1;
#20
reset = 0;
#10
start = 1'b1;
#10
start = 0;
#30
data_out0 = 16'd84;
#30
data_out0 = 16'd3;
```

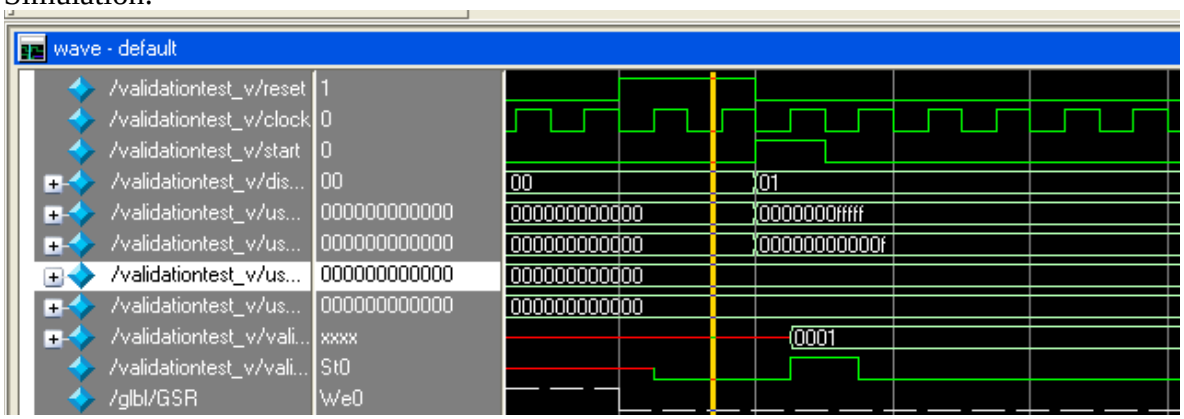
Simulation:



Validation:
Code:

```
reset = 1'b1;
#20
reset = 0;
start = 1'b1;
dis_count = 2'b01;
user0 = 48'h0000_0000_000f_ffff;
user1 = 48'h0000_0000_0000_000f;
user2 = 48'h0000_0000_0000_0000;
user3 = 48'h0000_0000_0000_0000;
#10
start = 0;
```

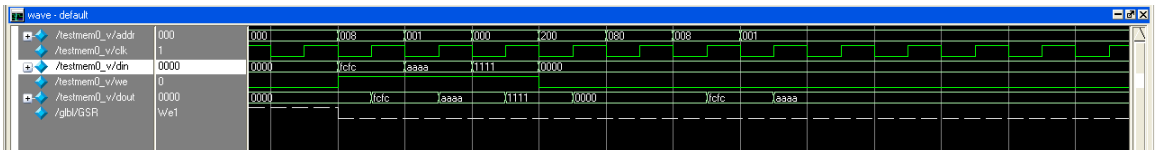
Simulation:



Mem Test:
Code:

```
addr = 10'b00_0000_1000;
din = 16'hfcfc;
we = 1'b1;
#10;
addr = 10'b00_0000_0001;
din = 16'haaaa;
we = 1'b1;
#10;
addr = 10'b00_0000_0000;
din = 16'h1111;
we = 1'b1;
#10
addr = 10'b10_0000_0000;
din = 16'h0000;
we = 1'b0;
#10
addr = 10'b00_1000_0000;
din = 16'h0000;
we = 1'b0;
#10
addr = 10'b00_0000_1000;
din = 16'h0000;
we = 1'b0;
#10
addr = 10'b00_0000_0001;
din = 16'h0000;
we = 1'b0;
```

Simuation:



5.2 Appendix : Code

Labkit Code:
Provided by 6.111, modified by Raiza Muñiz

```
//////////////////////////////////////////////////////////////////////////////
//
// 6.111 FPGA Labkit -- Lab 4: Pong
//
//
// Created: March 15, 2007
// Author: Nathan Ickes
//
// This is a template for implementing the Pong game for Lab 4. This file
// includes two modules:
//
//   - labkit: the top-level labkit module
//   - debounce: the synchronize/debounce module
//
// Students should modify and add modules according to the directions outlined
// in the lab 4 manual.
//
//////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////
//
// 6.111 FPGA Labkit -- Template Toplevel Module for Lab 4 (Spring 2007)
//
//
// Created: March 15, 2007
// Author: Nathan Ickes
//
//////////////////////////////////////////////////////////////////////////////

module labkit (beep, audio_reset_b, ac97_sdata_out, ac97_sdata_in, ac97_synch,
               ac97_bit_clock,

               vga_out_red, vga_out_green, vga_out_blue, vga_out_sync_b,
               vga_out_blank_b, vga_out_pixel_clock, vga_out_hsync,
               vga_out_vsync,

               tv_out_ycrcb, tv_out_reset_b, tv_out_clock, tv_out_i2c_clock,
               tv_out_i2c_data, tv_out_pal_ntsc, tv_out_hsync_b,
               tv_out_vsync_b, tv_out_blank_b, tv_out_subcar_reset,

               tv_in_ycrcb, tv_in_data_valid, tv_in_line_clock1,
               tv_in_line_clock2, tv_in_aef, tv_in_hff, tv_in_aff,
```

```
        tv_in_i2c_clock, tv_in_i2c_data, tv_in_fifo_read,
        tv_in_fifo_clock, tv_in_iso, tv_in_reset_b, tv_in_clock,

        ram0_data, ram0_address, ram0_adv_ld, ram0_clk, ram0_cen_b,
        ram0_ce_b, ram0_oe_b, ram0_we_b, ram0_bwe_b,

        ram1_data, ram1_address, ram1_adv_ld, ram1_clk, ram1_cen_b,
        ram1_ce_b, ram1_oe_b, ram1_we_b, ram1_bwe_b,

        clock_feedback_out, clock_feedback_in,

        flash_data, flash_address, flash_ce_b, flash_oe_b, flash_we_b,
        flash_reset_b, flash_sts, flash_byte_b,

        rs232_txd, rs232_rxd, rs232_rts, rs232_cts,

        mouse_clock, mouse_data, keyboard_clock, keyboard_data,

        clock_27mhz, clock1, clock2,

        disp_blank, disp_data_out, disp_clock, disp_rs, disp_ce_b,
        disp_reset_b, disp_data_in,

        button0, button1, button2, button3, button_enter, button_right,
        button_left, button_down, button_up,

        switch,

        led,

        user1, user2, user3, user4,

        daughtercard,

        systemace_data, systemace_address, systemace_ce_b,
        systemace_we_b, systemace_oe_b, systemace_irq, systemace_mpbrdy,

        analyzer1_data, analyzer1_clock,
        analyzer2_data, analyzer2_clock,
        analyzer3_data, analyzer3_clock,
        analyzer4_data, analyzer4_clock);

output beep, audio_reset_b, ac97_synch, ac97_sdata_out;
input  ac97_bit_clock, ac97_sdata_in;

output [7:0] vga_out_red, vga_out_green, vga_out_blue;
```

```
output vga_out_sync_b, vga_out_blank_b, vga_out_pixel_clock,
      vga_out_hsync, vga_out_vsync;

output [9:0] tv_out_ycrcb;
output tv_out_reset_b, tv_out_clock, tv_out_i2c_clock, tv_out_i2c_data,
      tv_out_pal_ntsc, tv_out_hsync_b, tv_out_vsync_b, tv_out_blank_b,
      tv_out_subcar_reset;

input  [19:0] tv_in_ycrcb;
input  tv_in_data_valid, tv_in_line_clock1, tv_in_line_clock2, tv_in_aef,
      tv_in_hff, tv_in_aff;
output tv_in_i2c_clock, tv_in_fifo_read, tv_in_fifo_clock, tv_in_iso,
      tv_in_reset_b, tv_in_clock;
inout  tv_in_i2c_data;

inout  [35:0] ram0_data;
output [18:0] ram0_address;
output ram0_adv_ld, ram0_clk, ram0_cen_b, ram0_ce_b, ram0_oe_b, ram0_we_b;
output [3:0] ram0_bwe_b;

inout  [35:0] ram1_data;
output [18:0] ram1_address;
output ram1_adv_ld, ram1_clk, ram1_cen_b, ram1_ce_b, ram1_oe_b, ram1_we_b;
output [3:0] ram1_bwe_b;

input  clock_feedback_in;
output clock_feedback_out;

inout  [15:0] flash_data;
output [23:0] flash_address;
output flash_ce_b, flash_oe_b, flash_we_b, flash_reset_b, flash_byte_b;
input  flash_sts;

output rs232_txd, rs232_rts;
input  rs232_rxd, rs232_cts;

input  mouse_clock, mouse_data, keyboard_clock, keyboard_data;

input  clock_27mhz, clock1, clock2;

output disp_blank, disp_clock, disp_rs, disp_ce_b, disp_reset_b;
input  disp_data_in;
output  disp_data_out;

input  button0, button1, button2, button3, button_enter, button_right,
      button_left, button_down, button_up;
```

```
input  [7:0] switch;
output [7:0] led;

inout [31:0] user1, user2, user3, user4;

inout [43:0] daughtercard;

inout  [15:0] systemace_data;
output [6:0]  systemace_address;
output systemace_ce_b, systemace_we_b, systemace_oe_b;
input  systemace_irq, systemace_mpbrdy;

output [15:0] analyzer1_data, analyzer2_data, analyzer3_data,
            analyzer4_data;
output analyzer1_clock, analyzer2_clock, analyzer3_clock, analyzer4_clock;

//////////////////////////////////////////////////////////////////////////
//
// I/O Assignments
//
//////////////////////////////////////////////////////////////////////////

// Audio Input and Output
assign beep= 1'b0;
assign audio_reset_b = 1'b0;
assign ac97_synch = 1'b0;
assign ac97_sdata_out = 1'b0;

// Video Output
assign tv_out_ycrcb = 10'h0;
assign tv_out_reset_b = 1'b0;
assign tv_out_clock = 1'b0;
assign tv_out_i2c_clock = 1'b0;
assign tv_out_i2c_data = 1'b0;
assign tv_out_pal_ntsc = 1'b0;
assign tv_out_hsync_b = 1'b1;
assign tv_out_vsync_b = 1'b1;
assign tv_out_blank_b = 1'b1;
assign tv_out_subcar_reset = 1'b0;

// Video Input
assign tv_in_i2c_clock = 1'b0;
assign tv_in_fifo_read = 1'b0;
assign tv_in_fifo_clock = 1'b0;
assign tv_in_iso = 1'b0;
assign tv_in_reset_b = 1'b0;
```

```
assign tv_in_clock = 1'b0;
assign tv_in_i2c_data = 1'bZ;

// SRAMs
assign ram0_data = 36'hZ;
assign ram0_address = 19'h0;
assign ram0_adv_ld = 1'b0;
assign ram0_clk = 1'b0;
assign ram0_cen_b = 1'b1;
assign ram0_ce_b = 1'b1;
assign ram0_oe_b = 1'b1;
assign ram0_we_b = 1'b1;
assign ram0_bwe_b = 4'hF;
assign ram1_data = 36'hZ;
assign ram1_address = 19'h0;
assign ram1_adv_ld = 1'b0;
assign ram1_clk = 1'b0;
assign ram1_cen_b = 1'b1;
assign ram1_ce_b = 1'b1;
assign ram1_oe_b = 1'b1;
assign ram1_we_b = 1'b1;
assign ram1_bwe_b = 4'hF;
assign clock_feedback_out = 1'b0;

// Flash ROM
assign flash_data = 16'hZ;
assign flash_address = 24'h0;
assign flash_ce_b = 1'b1;
assign flash_oe_b = 1'b1;
assign flash_we_b = 1'b1;
assign flash_reset_b = 1'b0;
assign flash_byte_b = 1'b1;

// RS-232 Interface
assign rs232_txd = 1'b1;
assign rs232_rts = 1'b1;

// LED Displays
assign disp_blank = 1'b1;
assign disp_clock = 1'b0;
assign disp_rs = 1'b0;
assign disp_ce_b = 1'b1;
assign disp_reset_b = 1'b0;
assign disp_data_out = 1'b0;

// Buttons, Switches, and Individual LEDs
```

```
//assign led = 8'hFF;

// User I/Os
assign user1 = 32'hZ;
assign user2 = 32'hZ;
assign user3 = 32'hZ;
assign user4 = 32'hZ;

// Daughtercard Connectors
assign daughtercard = 44'hZ;

// SystemACE Microprocessor Port
assign systemace_data = 16'hZ;
assign systemace_address = 7'h0;
assign systemace_ce_b = 1'b1;
assign systemace_we_b = 1'b1;
assign systemace_oe_b = 1'b1;

// Logic Analyzer
assign analyzer1_data = 16'h0;
assign analyzer1_clock = 1'b1;
assign analyzer2_data = 16'h0;
assign analyzer2_clock = 1'b1;
assign analyzer3_data = 16'h0;
assign analyzer3_clock = 1'b1;
assign analyzer4_data = 16'h0;
assign analyzer4_clock = 1'b1;

////////////////////////////////////////////////////////////////////////
//
// Lab 4 Components
//
////////////////////////////////////////////////////////////////////////

//
// Generate a 31.5MHz pixel clock from clock_27mhz
//

wire pclk, pixel_clock;
DCM pixel_clock_dcm (.CLKIN(clock_27mhz), .CLKFX(pclk));
// synthesis attribute CLKFX_DIVIDE of pixel_clock_dcm is 6
// synthesis attribute CLKFX_MULTIPLY of pixel_clock_dcm is 7
// synthesis attribute CLK_FEEDBACK of pixel_clock_dcm  is "NONE"
BUFG pixel_clock_buf (.I(pclk), .O(pixel_clock));

//
```

```
  // VGA output signals
  //

  // Inverting the clock to the DAC provides half a clock period for signals
  // to propagate from the FPGA to the DAC.
  assign vga_out_pixel_clock = ~pixel_clock;

  // The composite sync signal is used to encode sync data in the green
  // channel analog voltage for older monitors.  It does not need to be
  // implemented for the monitors in the 6.111 lab, and can be left at 1'b1.
  assign vga_out_sync_b = 1'b1;

  // The following assignments should be deleted and replaced with your own
  // code to implement the Pong game.
  //assign vga_out_red = 8'h0;
  //assign vga_out_green = 8'h0;
  //assign vga_out_blue = 8'h0;
  //assign vga_out_blank_b = 1'b1;
  //assign vga_out_hsync = 1'b0;
  //assign vga_out_vsync = 1'b0;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
//  Final Project
//
//
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////Debounce Signals//////////////////////////////////////////////////////////
wire reset_sync, add_sync, id_sync;
debounce debouncereset(0, pixel_clock, ~button0, reset_sync);
debounce debouncadd(reset_sync, pixel_clock, ~button2, add_sync);
debounce debounceid(reset_sync, pixel_clock, ~button3, id_sync);

//////////////////////////////////////////////Control Unit//////////////////////////////////////////////////////////////
wire id_start, add_start, validation_start, extract;
wire allusers_used;
wire reset_reg;
wire writee;
wire id_busy, add_busy, validation_busy, validation_done, extracted;
wire enablew, enabled;
wire [1:0] action;
wire [1:0] user;
wire [3:0] validated_user;
wire [3:0] vgaoutput;
```

```
wire [7:0] led;
wire [15:0] distance;
wire [15:0] feat_comp0, feat_comp1, feat_comp2, feat_comp3, feat_comp4,
feat_comp5, feat_comp6;
wire [15:0] feat_comp7, feat_comp8, feat_comp9, feat_comp10, feat_comp11,
feat_comp12, feat_comp13, feat_comp14, feat_comp15;


controlunit2 controlunit2222(reset_sync, pixel_clock, extracted, action,
distance, extract, extraction_done, add_start, id_start,
        validation_start, user, allusers_used, vgaoutput, reset_reg,
led);

action_reg actionreg(pixel_clock, reset_sync, reset_reg, add_sync, id_sync,
action);

//////////////////////////////////////////////////////Memory//////////////////////////////////////////////////////////////

mem mem0(.addr(address), .clk(pixel_clock), .din(data_in0), .dout(data_out0),
.we(writee));
mem mem1(.addr(address), .clk(pixel_clock), .din(data_in1), .dout(data_out1),
.we(writee));
mem mem2(.addr(address), .clk(pixel_clock), .din(data_in2), .dout(data_out2),
.we(writee));
mem mem3(.addr(address), .clk(pixel_clock), .din(data_in3), .dout(data_out3),
.we(writee));
mem mem4(.addr(address), .clk(pixel_clock), .din(data_in4), .dout(data_out4),
.we(writee));
mem mem5(.addr(address), .clk(pixel_clock), .din(data_in5), .dout(data_out5),
.we(writee));
mem mem6(.addr(address), .clk(pixel_clock), .din(data_in6), .dout(data_out6),
.we(writee));
mem mem7(.addr(address), .clk(pixel_clock), .din(data_in7), .dout(data_out7),
.we(writee));
mem mem8(.addr(address), .clk(pixel_clock), .din(data_in8), .dout(data_out8),
.we(writee));
mem mem9(.addr(address), .clk(pixel_clock), .din(data_in9), .dout(data_out9),
.we(writee));
mem mem10(.addr(address), .clk(pixel_clock), .din(data_in10), .dout(data_out10),
.we(writee));
mem mem11(.addr(address), .clk(pixel_clock), .din(data_in11), .dout(data_out11),
.we(writee));
mem mem12(.addr(address), .clk(pixel_clock), .din(data_in12), .dout(data_out12),
.we(writee));
mem mem13(.addr(address), .clk(pixel_clock), .din(data_in13), .dout(data_out13),
.we(writee));
```

```
mem mem14(.addr(address), .clk(pixel_clock), .din(data_in14), .dout(data_out14),
.we(writee));
mem mem15(.addr(address), .clk(pixel_clock), .din(data_in15), .dout(data_out15),
.we(writee));


//////////////////////////////////////////////////////Display//////////////////////////////////////////////////////////

wire hsync;
wire vsync;
wire [9:0] pixel_count;
wire [8:0] line_count;
wire hblank, vblank;
wire hsync_delay, vsync_delay;

assign vga_out_blank_b = (hblank & vblank);
assign vga_out_hsync = hsync_delay;
assign vga_out_vsync = vsync_delay;

delay delay1(pixel_clock, hsync, vsync, hsync_delay, vsync_delay);

vga vga1(pixel_clock, reset_sync, hsync, vsync, hblank, vblank, pixel_count,
line_count);

displayfield displayfield1(pixel_count, line_count, vgaoutput, vga_out_red,
vga_out_green, vga_out_blue);

endmodule

//////////////////////////////////////////////////////////////////////////////
//
// 6.111 FPGA Labkit -- Debounce/Synchronize module
//
//
// Use your system clock for the clock input to produce a synchronous,
// debounced output
//
//////////////////////////////////////////////////////////////////////////////

module debounce (reset, clock, noisy, clean);
  parameter DELAY = 270000;   // .01 sec with a 27Mhz clock
  input reset, clock, noisy;
  output clean;

  reg [18:0] count;
  reg new, clean;
```

```
  always @(posedge clock)
    if (reset)
      begin
        count <= 0;
        new <= noisy;
        clean <= noisy;
      end
    else if (noisy != new)
      begin
        new <= noisy;
        count <= 0;
      end
    else if (count == DELAY)
      clean <= new;
    else
      count <= count+1;

endmodule
```

Control Unit:
By Raiza Muñiz

```
module controlunit2(reset, clock, extracted, action, distance, extract, extraction_done,
add_start, id_start, validation_start, user, allusers_used, vgaoutput, reset_reg,leds);

input reset, clock, extracted;
input [1:0] action;
input [15:0] distance;

output id_start, add_start, validation_start, reset_reg;
output extract, allusers_used, extraction_done;
output [2:0] user;
output [3:0] vgaoutput;
output [7:0] leds;

reg id_start, add_start, validation_start, reset_reg;
reg extract, allusers_used, extraction_done;
reg [2:0] user;
reg [3:0] vgaoutput;
reg [7:0] leds;
reg [2:0] state, next_state;

wire [15:0] data_out0, data_out1, data_out2, data_out3, data_out4, data_out5,
data_out6, data_out7;
wire [15:0] data_out8, data_out9, data_out10, data_out11, data_out12,
```

```
data_out13, data_out14,  data_out15;
wire [15:0] feature0, feature1,feature2, feature3, feature4, feature5, feature6,
feature7;
wire [15:0] feature8, feature9, feature10, feature11, feature12, feature13,
feature14, feature15;
wire [47:0] user0, user1, user2, user3;
wire writee, enablew;
wire [10:0] address;
wire [15:0] feat_comp0, feat_comp1, feat_comp2, feat_comp3, feat_comp4,
feat_comp5, feat_comp6;
wire [15:0] feat_comp7, feat_comp8, feat_comp9, feat_comp10, feat_comp11,
feat_comp12, feat_comp13, feat_comp14, feat_comp15;

wire add_busy, id_busy;
wire enablew_add, enablew_id, writee_add, writee_id, enabled;
wire validation_done;
wire [1:0] dis_count;
wire [3:0] validated_user;
wire [10:0] address_add, address_id;
wire [15:0] data_in0, data_in1, data_in2, data_in3, data_in4, data_in5,
data_in6, data_in7;
wire [15:0] data_in8, data_in9, data_in10, data_in11, data_in12, data_in13,
data_in14, data_in15;

parameter RESET = 0;
parameter IDLE = 3;
parameter ADD = 1;
parameter ID = 2;
parameter BUSY0 = 4;
parameter BUSY1 = 5;
parameter BUSY2 = 6;

always @(posedge clock)
begin
      if (reset)
      state <= RESET;
      else state <= next_state;
end

always @(state or extracted or action or add_busy or id_busy or validation_done)
begin
   case (state)
            RESET: begin
                  extract <= 0;
                  extraction_done <= 0;
                  add_start <= 0;
```

```
            id_start <= 0;
            validation_start <= 0;
            user <= 0;
            allusers_used <= 0;
            vgaoutput <= 4'b1011;
            reset_reg <= 0;
            leds <= 8'b0000_0000;
            next_state <= IDLE;
            end

IDLE:
      if (action == ADD || action == ID)
      if (~extraction_done)  //haven't extracted
       begin
       extract <= 1'b1;
       extraction_done <= extraction_done;
       add_start <= 0;
       id_start <= 0;
       validation_start <= 0;
       vgaoutput <= 4'b1000;
       reset_reg <= 1'b0;
       user <= user;
       allusers_used <= allusers_used;
       leds <= 8'b1111_1111;
       next_state <= BUSY0;
       end

   else //extraction done - perform action
    if (action == ADD) //ADD
    begin
       extract <= 0;
        extraction_done <= extraction_done;
       add_start <= 1'b1;  //activate add
       id_start <= 0;
       validation_start <= 0;
       vgaoutput <= 4'b1101;  //ADD
       reset_reg <= 1'b0;
       user <= user;
       allusers_used <= allusers_used;
       leds <= 8'b1111_1101;  //ADD
       next_state <= BUSY1;
    end
    else
    begin
       extract <= 0;
       extraction_done <= extraction_done;
```

```
    add_start <= 0;
    id_start <= 1'b1;  //activate id
    validation_start <= 1'b1;
    user <= user;
    allusers_used <= allusers_used;
    vgaoutput <= 4'b1100;
    reset_reg <= 1'b0;
    leds <= 8'b1111_1011;
    next_state <= BUSY2;
end

else // no action
 begin
    extract <= 0;
    extraction_done <= 0;
    add_start <= 0;
    id_start <= 0;
    validation_start <= 0;
    user <= user;
    allusers_used <= allusers_used;
    vgaoutput <= 4'b1011;
    reset_reg <= 0;
    leds <= 8'b1111_1110;
    next_state <= IDLE;
end

      BUSY0: begin
      if (~extracted)
        begin
        extract <= 0;
        extraction_done <= 0;
    add_start <= 0;
    id_start <= 0;  //activate id
    validation_start <= 0;
    user <= user;
    allusers_used <= allusers_used;
    vgaoutput <= 4'b1000;
    reset_reg <= 1'b0;
    leds <= 8'b1111_0000;
    next_state <= BUSY0;
          end
          else //extracted
          begin
          extract <= 0;
         extraction_done <= 1'b1;
    add_start <= 0;
```

```
        id_start <= 0;  //activate id
        validation_start <= 0;
        user <= user;
        allusers_used <= allusers_used;
        vgaoutput <= 4'b1001;
        reset_reg <= 1'b0;
        leds <= 8'b0111_1111;
        next_state <= IDLE;
              end
              end

         BUSY1: begin  //for ADD
    if (add_busy)
    begin
        extract <= 0;
         extraction_done <= extraction_done;
        add_start <= 0;
        id_start <= 0;
        validation_start <= 0;
        user <= user;
        allusers_used <= allusers_used;
        vgaoutput <= 4'b1101;
        reset_reg <= 0;
        leds <= 8'b1111_1101;
        next_state <= BUSY1;
    end
    else //add done
    begin
         if (user == 2'b11) //=3  //needs to erase one of the users
        begin
            user <= 0;
            allusers_used <= 1'b1;
                 end
        else user <= user + 1;
        extract <= 0;
        extraction_done <= extraction_done;
        add_start <= 0;
        id_start <= 0;
        validation_start <= 0;
        allusers_used <= allusers_used;
        vgaoutput <= 4'b1010;  //ADD Done
        reset_reg <= 1'b1;
        leds <= 8'b1111_0111;
        next_state <= IDLE;
    end
end
```

```
BUSY2: begin
   if (id_busy)
   begin
      extract <= 0;
      extraction_done <= extraction_done;
      add_start <= 0;
      id_start <= 0;
      validation_start <= 0;
      user <= user;
      allusers_used <= allusers_used;
      vgaoutput <= 4'b1100;  //ID
      reset_reg <= 0;
      leds <= 8'b1111_1011;
      next_state <= BUSY2;
   end
   else //id mem done
   if (validation_done) //validated
   begin
      extract <= 0;
       extraction_done <= extraction_done;
       add_start <= 0;
      id_start <= 0;
      validation_start <= 0;
      user <= user;
      allusers_used <= allusers_used;
      vgaoutput <= {1'b0, validated_user[2:0]};
      reset_reg <= 1'b1;
      leds <= 8'b1111_1111;
      next_state <= IDLE;
                           end
   else //haven't validated yet
   begin
      extract <= 0;
      extraction_done <= extraction_done;
      add_start <= 0;
      id_start <= 0;
      validation_start <= 0;
      user <= user;
      allusers_used <= allusers_used;
      vgaoutput <= 4'b1110;  //validating
      reset_reg <= 0;
     leds <= 8'b1111_1011;
      next_state <= BUSY2;
   end
end
```

```
            default:  begin
                        user <= 0;
                        next_state <= IDLE;
                        extraction_done <= 0;
                        end

endcase
end

assign writee = (writee_add |writee_id);
assign address = (add_busy) ? address_add : (id_busy) ? address_id :
10'b00_0000_0000;

validation validate(reset, clock, validation_start, validated_user,
validation_done, dis_count, user0, user1, user2, user3);

mem_id id(clock, reset, id_start, user, enablew_id, enabled, writee_id,
address_id, allusers_used, distance, id_busy, data_out0, data_out1, data_out2, data_out3,
data_out4, data_out5, data_out6, data_out7, data_out8, data_out9, data_out10,
data_out11, data_out12, data_out13, data_out14,  data_out15, feat_comp0, feat_comp1,
feat_comp2, feat_comp3, feat_comp4, feat_comp5, feat_comp6, feat_comp7,
feat_comp8, feat_comp9, feat_comp10, feat_comp11, feat_comp12, feat_comp13,
feat_comp14, feat_comp15);

mem_add add(reset, clock, add_start, user, enablew_add, writee_add, address_add,
add_busy, feature0, feature1,feature2, feature3, feature4, feature5, feature6, feature7,
feature8, feature9, feature10, feature11, feature12, feature13, feature14, feature15,
data_in0, data_in1, data_in2, data_in3, data_in4, data_in5, data_in6, data_in7,
data_in8, data_in9, data_in10, data_in11, data_in12, data_in13, data_in14, data_in15);
 endmodule

MEM_ADD:
by Raiza Muñiz
module mem_add(clock, reset, start, user, enable, writee, address, busy,
feature0, feature1, feature2, feature3, feature4, feature5, feature6, feature7, feature8,
feature9, feature10, feature11, feature12, feature13, feature14, feature15, data_in0,
data_in1, data_in2, data_in3, data_in4, data_in5, data_in6, data_in7, data_in8,
data_in9, data_in10, data_in11, data_in12, data_in13, data_in14, data_in15);

input clock, reset, start;
input [1:0] user;
input [15:0] feature0, feature1, feature2, feature3, feature4, feature5,
feature6, feature7;
input [15:0] feature8, feature9, feature10, feature11, feature12, feature13,
feature14, feature15;
```

```
output enable, writee, busy;
output [10:0] address;
output [15:0] data_in0, data_in1, data_in2, data_in3, data_in4, data_in5,
data_in6, data_in7;
output [15:0] data_in8, data_in9, data_in10, data_in11, data_in12, data_in13,
data_in14, data_in15;


reg enable, writee, busy;
reg [1:0] state, next_state;
reg [11:0] count;
reg [10:0] address;
reg [15:0] data_in0, data_in1, data_in2, data_in3, data_in4, data_in5, data_in6,
data_in7;
reg [15:0] data_in8, data_in9, data_in10, data_in11, data_in12, data_in13,
data_in14, data_in15;

parameter IDLE = 0;
parameter PROCESSING1 = 1;
parameter PROCESSING2 = 2;
parameter TOTAL = 159;

always @(posedge clock)
if (reset)
  state <= IDLE;
else state <= next_state;

always @(state or start)
begin
case (state)
   IDLE: begin
      if (~reset)
   if (start)
   begin
      if (user == 0)
      begin
         busy <= 1'b1;
         enable <= 1'b1;  //enable memory
         writee <=  1'b1;//write
         address <= 10'b00_0000_0000; //needs to be modified
         count <= count + 1;
         next_state <= PROCESSING1;
                        data_in0 <= feature0;
                        data_in1 <= feature1;
                        data_in2 <= feature2;
                        data_in3 <= feature3;
```

```
                    data_in4 <= feature4;
                    data_in5 <= feature5;
                    data_in6 <= feature6;
                    data_in7 <= feature7;
                    data_in8 <= feature8;
                    data_in9 <= feature9;
                    data_in10 <= feature10;
                    data_in11 <= feature11;
                    data_in12 <= feature12;
                    data_in13 <= feature13;
                    data_in14 <= feature14;
                    data_in15 <= feature15;
            end
    else begin  //user !=0 - continue
            busy <= 1'b1;
            enable <= 1'b1;
            writee <=  1'b1;//write
            address <= address + 1;
            count <= count +1;
            next_state <= PROCESSING1;
                            data_in0 <= feature0;
                            data_in1 <= feature1;
                            data_in2 <= feature2;
                            data_in3 <= feature3;
                            data_in4 <= feature4;
                            data_in5 <= feature5;
                            data_in6 <= feature6;
                            data_in7 <= feature7;
                            data_in8 <= feature8;
                            data_in9 <= feature9;
                            data_in10 <= feature10;
                            data_in11 <= feature11;
                            data_in12 <= feature12;
                            data_in13 <= feature13;
                            data_in14 <= feature14;
                            data_in15 <= feature15;
        end
  end
else begin  //no start
    busy <= 0;
    enable <= 0;
    writee <= 0 ;//read
    address <= address;
    count <= count;
    next_state <= IDLE;
            data_in0 <= 16'h0000;
```

```
            data_in1 <= 16'h0000;
                data_in2 <= 16'h0000;
                data_in3 <= 16'h0000;
                data_in4 <= 16'h0000;
                data_in5 <= 16'h0000;
                data_in6 <= 16'h0000;
                data_in7 <= 16'h0000;
                data_in8 <= 16'h0000;
                data_in9 <= 16'h0000;
                data_in10 <= 16'h0000;
                data_in11 <= 16'h0000;
                data_in12 <= 16'h0000;
                data_in13 <= 16'h0000;
                data_in14 <= 16'h0000;
                data_in15 <= 16'h0000;
    end
      else //reset
      begin
          busy <= 0;
      enable <= 0;
      writee <= 0 ;//read
      address <= 10'b00_0000_0000;
      count <= 0;
      next_state <= IDLE;
            data_in0 <= 16'h0000;
            data_in1 <= 16'h0000;
                data_in2 <= 16'h0000;
                data_in3 <= 16'h0000;
                data_in4 <= 16'h0000;
                data_in5 <= 16'h0000;
                data_in6 <= 16'h0000;
                data_in7 <= 16'h0000;
                data_in8 <= 16'h0000;
                data_in9 <= 16'h0000;
                data_in10 <= 16'h0000;
                data_in11 <= 16'h0000;
                data_in12 <= 16'h0000;
                data_in13 <= 16'h0000;
                data_in14 <= 16'h0000;
                data_in15 <= 16'h0000;
            end
    end

    PROCESSING1:
  begin
    if (count == TOTAL)
```

```
begin
   busy <= 0;
   enable <= 0;
   writee <= 0;//read
   address <= address;
   count <= 0;  //reset
   next_state <= IDLE;
            data_in0 <= 16'h0000;
            data_in1 <= 16'h0000;
                 data_in2 <= 16'h0000;
                 data_in3 <= 16'h0000;
                 data_in4 <= 16'h0000;
                 data_in5 <= 16'h0000;
                 data_in6 <= 16'h0000;
                 data_in7 <= 16'h0000;
                 data_in8 <= 16'h0000;
                 data_in9 <= 16'h0000;
                 data_in10 <= 16'h0000;
                 data_in11 <= 16'h0000;
                 data_in12 <= 16'h0000;
                 data_in13 <= 16'h0000;
                 data_in14 <= 16'h0000;
                 data_in15 <= 16'h0000;
   end
else //(count != TOTAL)
begin
   busy <= 1'b1;
   enable <= 1'b1;
   writee <=  1'b1;//write
   address <= address + 1; //
   count <= count + 1;
   next_state <= PROCESSING2;
            data_in0 <= feature0;
            data_in1 <= feature1;
                 data_in2 <= feature2;
                 data_in3 <= feature3;
                 data_in4 <= feature4;
                 data_in5 <= feature5;
                 data_in6 <= feature6;
                 data_in7 <= feature7;
             data_in8 <= feature8;
                 data_in9 <= feature9;
                 data_in10 <= feature10;
                 data_in11 <= feature11;
                 data_in12 <= feature12;
                 data_in13 <= feature13;
```

```
                data_in14 <= feature14;
                data_in15 <= feature15;
    end
end

    PROCESSING2: begin
    if (count == TOTAL)
    begin
        busy <= 0;
        enable <= 0;
        writee <= 0;//read
        address <= address;
        count <= 0;  //reset
        next_state <= IDLE;
                data_in0 <= 16'h0000;
                data_in1 <= 16'h0000;
                    data_in2 <= 16'h0000;
                    data_in3 <= 16'h0000;
                    data_in4 <= 16'h0000;
                    data_in5 <= 16'h0000;
                    data_in6 <= 16'h0000;
                    data_in7 <= 16'h0000;
                    data_in8 <= 16'h0000;
                    data_in9 <= 16'h0000;
                    data_in10 <= 16'h0000;
                    data_in11 <= 16'h0000;
                    data_in12 <= 16'h0000;
                    data_in13 <= 16'h0000;
                    data_in14 <= 16'h0000;
                    data_in15 <= 16'h0000;
        end
    else //(count != TOTAL)
    begin
        busy <= 1'b1;
        enable <= 1'b1;
        writee <=  1'b1;//write
        address <= address + 1; //
        count <= count + 1;
        next_state <= PROCESSING1;
                data_in0 <= feature0;
                data_in1 <= feature1;
                    data_in2 <= feature2;
                    data_in3 <= feature3;
                    data_in4 <= feature4;
                    data_in5 <= feature5;
                    data_in6 <= feature6;
```

```
                data_in7 <= feature7;
            data_in8 <= feature8;
                data_in9 <= feature9;
                data_in10 <= feature10;
                data_in11 <= feature11;
                data_in12 <= feature12;
                data_in13 <= feature13;
                data_in14 <= feature14;
                data_in15 <= feature15;
   end
end

default: begin
                address <= 10'b00_0000_0000;
                count <= 0;
                end

endcase
end
endmodule
```

MEM_ID:
by Raiza Muñiz

```
module mem_id(clock, reset, start, user, enablew, enabled, writee, address,
allusers_used, distance, dis_count, busy, data_out0, data_out1, data_out2, data_out3,
data_out4, data_out5, data_out6, data_out7,  data_out8, data_out9, data_out10,
data_out11, data_out12, data_out13, data_out14,  data_out15,  feat_comp0, feat_comp1,
feat_comp2, feat_comp3, feat_comp4, feat_comp5, feat_comp6,  feat_comp7,
feat_comp8, feat_comp9, feat_comp10, feat_comp11, feat_comp12, feat_comp13,
feat_comp14, feat_comp15);

input clock, reset, start, allusers_used;
input [1:0] user;
input [15:0] distance;
input [15:0] data_out0, data_out1, data_out2, data_out3, data_out4, data_out5,
data_out6, data_out7;
input [15:0] data_out8, data_out9, data_out10, data_out11, data_out12,
data_out13, data_out14,  data_out15;

output enablew, enabled, writee, busy;
output [1:0] dis_count;
output [10:0] address;
output [15:0] feat_comp0, feat_comp1, feat_comp2, feat_comp3, feat_comp4,
feat_comp5, feat_comp6, feat_comp7;
output [15:0] feat_comp8, feat_comp9, feat_comp10, feat_comp11, feat_comp12,
feat_comp13, feat_comp14, feat_comp15;
```

```
reg busy, enablew, enabled, writee;
reg [1:0] dis_count;
reg [1:0] state, next_state;
reg [10:0] address;
reg [11:0] count;
reg [1:0] intuser_count;
reg [15:0] feat_comp0, feat_comp1, feat_comp2, feat_comp3, feat_comp4,
feat_comp5, feat_comp6;
reg [15:0] feat_comp7, feat_comp8, feat_comp9, feat_comp10, feat_comp11,
feat_comp12, feat_comp13, feat_comp14, feat_comp15;

parameter IDLE = 0;
parameter PROCESSING1 = 1;
parameter PROCESSING2 = 2;
parameter TOTAL = 159;

always @(posedge clock)
if (reset)
  state <= IDLE;
else
state <= next_state;

always @(state or start or data_out0 or data_out1 or data_out2 or data_out3 or
data_out4 or data_out5 or data_out6 or data_out7 or
                data_out8 or data_out9 or data_out10 or data_out11 or data_out12 or
data_out13 or data_out14 or  data_out15)
      begin
   case(state)
      IDLE: begin
              if (~reset)
         if (start)
         begin
            busy <= 1'b1;
            enablew <= 1'b1;  //begin reading
            enabled <= 0;
            writee <=  0;  //read
            address <= 10'b00_0000_0000;
            count <= count + 1;
            intuser_count <= intuser_count - 1;
            next_state <= PROCESSING1;
                           feat_comp0 <= 16'h0000;
                           feat_comp1 <= 16'h0000;
                           feat_comp2 <= 16'h0000;
                           feat_comp3 <= 16'h0000;
                           feat_comp4 <= 16'h0000;
```

```
                    feat_comp5 <= 16'h0000;
                    feat_comp6 <= 16'h0000;
                    feat_comp7 <= 16'h0000;
                    feat_comp8 <= 16'h0000;
                    feat_comp9 <= 16'h0000;
                    feat_comp10 <= 16'h0000;
                    feat_comp11 <= 16'h0000;
                    feat_comp12 <= 16'h0000;
                    feat_comp13 <= 16'h0000;
                    feat_comp14 <= 16'h0000;
                    feat_comp15 <= 16'h0000;

end

   else begin  //no action
   busy <= 0;
   enablew <= 0;
   enabled <= 0;
   writee <=  0;  //read
   address <= address;
   count <= 0;
   intuser_count <= intuser_count;
   next_state <= IDLE;
                    feat_comp0 <= 16'h0000;
                    feat_comp1 <= 16'h0000;
                    feat_comp2 <= 16'h0000;
                    feat_comp3 <= 16'h0000;
                    feat_comp4 <= 16'h0000;
                    feat_comp5 <= 16'h0000;
                    feat_comp6 <= 16'h0000;
                    feat_comp7 <= 16'h0000;
                    feat_comp8 <= 16'h0000;
                    feat_comp9 <= 16'h0000;
                    feat_comp10 <= 16'h0000;
                    feat_comp11 <= 16'h0000;
                    feat_comp12 <= 16'h0000;
                    feat_comp13 <= 16'h0000;
                    feat_comp14 <= 16'h0000;
                    feat_comp15 <= 16'h0000;
   end

            else // reset
            begin
                busy <= 0;
   enablew <= 0;
   enabled <= 0;
```

```
            writee <=  0;  //read
            address <= 10'b00_0000_0000;
            count <= 0;
            intuser_count <= intuser_count;
            next_state <= IDLE;
                              feat_comp0 <= 16'h0000;
                              feat_comp1 <= 16'h0000;
                              feat_comp2 <= 16'h0000;
                              feat_comp3 <= 16'h0000;
                              feat_comp4 <= 16'h0000;
                              feat_comp5 <= 16'h0000;
                              feat_comp6 <= 16'h0000;
                              feat_comp7 <= 16'h0000;
                              feat_comp8 <= 16'h0000;
                              feat_comp9 <= 16'h0000;
                              feat_comp10 <= 16'h0000;
                              feat_comp11 <= 16'h0000;
                              feat_comp12 <= 16'h0000;
                              feat_comp13 <= 16'h0000;
                              feat_comp14 <= 16'h0000;
                              feat_comp15 <= 16'h0000;
            end
                        end

        PROCESSING1: begin
          if (data_out0 != 0 && data_out1 != 0 && data_out2 != 0 && data_out3
!= 0 && data_out4 != 0 && data_out5 != 0 && data_out6 != 0 && data_out7 != 0 &&
data_out8 != 0)  //got data
          if (count == TOTAL)  //got all data for 1 user
          if (intuser_count == 0)  //last user
          begin
            busy <= 0;
            enablew <= 0;
            enabled <= 1'b1;  //enable validation
            writee <= 0;
            address <= 10'b00_0000_0000;
            count <= 0;  //reset
            intuser_count <= intuser_count;
            next_state <= IDLE; ///pass data to Distance module
                              feat_comp0 <= data_out0;
                              feat_comp1 <= data_out1;
                              feat_comp2 <= data_out2;
                              feat_comp3 <= data_out3;
                              feat_comp4 <= data_out4;
                              feat_comp5 <= data_out5;
                              feat_comp6 <= data_out6;
```

```
                        feat_comp7 <= data_out7;
                        feat_comp8 <= data_out8;
                        feat_comp9 <= data_out9;
                        feat_comp10 <= data_out10;
                        feat_comp11 <= data_out11;
                        feat_comp12 <= data_out12;
                        feat_comp13 <= data_out13;
                        feat_comp14 <= data_out14;
                        feat_comp15 <= data_out15;
        end
        else // intuser-count != 0 //one more user tested, bring the next
one
        begin
           busy <= 1'b1;
           enablew <= 1'b1;
           enabled <= 1'b1;
           writee <= 0;  //read
           address <= address + 1;
           count <= count + 1;
           intuser_count <= intuser_count - 1;
           next_state <= PROCESSING2;
                        feat_comp0 <= data_out0;
                        feat_comp1 <= data_out1;
                        feat_comp2 <= data_out2;
                        feat_comp3 <= data_out3;
                        feat_comp4 <= data_out4;
                        feat_comp5 <= data_out5;
                        feat_comp6 <= data_out6;
                        feat_comp7 <= data_out7;
                        feat_comp8 <= data_out8;
                        feat_comp9 <= data_out9;
                        feat_comp10 <= data_out10;
                        feat_comp11 <= data_out11;
                        feat_comp12 <= data_out12;
                        feat_comp13 <= data_out13;
                        feat_comp14 <= data_out14;
                        feat_comp15 <= data_out15;
        end
        else //count != Total // haven't completed testing a user
        begin
           busy <= 1'b1;
           enablew <= 1'b1;
           enabled <= 1'b1;
           writee <= 0;  //read
           address <= address + 1;
           count <= count + 1;
```

```
    intuser_count <= intuser_count;
    next_state <= PROCESSING2;
                        feat_comp0 <= data_out0;
                        feat_comp1 <= data_out1;
                        feat_comp2 <= data_out2;
                        feat_comp3 <= data_out3;
                        feat_comp4 <= data_out4;
                        feat_comp5 <= data_out5;
                        feat_comp6 <= data_out6;
                        feat_comp7 <= data_out7;
                        feat_comp8 <= data_out8;
                        feat_comp9 <= data_out9;
                        feat_comp10 <= data_out10;
                        feat_comp11 <= data_out11;
                        feat_comp12 <= data_out12;
                        feat_comp13 <= data_out13;
                        feat_comp14 <= data_out14;
                        feat_comp15 <= data_out15;
end
else //new data hasn't arrived
begin
    busy <= 1'b1;
    enablew <= 1'b1;
    enabled <= 0;
    writee <= 0;  //read
    address <= address;
    count <= count;
    intuser_count <= intuser_count;
    next_state <= PROCESSING2;
                        feat_comp0 <= data_out0;
                        feat_comp1 <= data_out1;
                        feat_comp2 <= data_out2;
                        feat_comp3 <= data_out3;
                        feat_comp4 <= data_out4;
                        feat_comp5 <= data_out5;
                        feat_comp6 <= data_out6;
                        feat_comp7 <= data_out7;
                        feat_comp8 <= data_out8;
                        feat_comp9 <= data_out9;
                        feat_comp10 <= data_out10;
                        feat_comp11 <= data_out11;
                        feat_comp12 <= data_out12;
                        feat_comp13 <= data_out13;
                        feat_comp14 <= data_out14;
                        feat_comp15 <= data_out15;
end
```

```
        end

              PROCESSING2: begin
           if (data_out0 != 0 && data_out1 != 0 && data_out2 != 0 && data_out3
!= 0 && data_out4 != 0 && data_out5 != 0 && data_out6 != 0 && data_out7 != 0 &&
data_out8 != 0)  //got data
           if (count == TOTAL)  //got all data for 1 user
           if (intuser_count == 0)  //last user
           begin
              busy <= 0;
              enablew <= 0;
              enabled <= 1'b1;  //enable validation
              writee <= 0;
              address <= 10'b00_0000_0000;
              count <= 0;  //reset
              intuser_count <= intuser_count;
              next_state <= IDLE; ///pass data to Distance module
                                  feat_comp0 <= data_out0;
                                  feat_comp1 <= data_out1;
                                  feat_comp2 <= data_out2;
                                  feat_comp3 <= data_out3;
                                  feat_comp4 <= data_out4;
                                  feat_comp5 <= data_out5;
                                  feat_comp6 <= data_out6;
                                  feat_comp7 <= data_out7;
                                  feat_comp8 <= data_out8;
                                  feat_comp9 <= data_out9;
                                  feat_comp10 <= data_out10;
                                  feat_comp11 <= data_out11;
                                  feat_comp12 <= data_out12;
                                  feat_comp13 <= data_out13;
                                  feat_comp14 <= data_out14;
                                  feat_comp15 <= data_out15;
           end
           else // intuser-count != 0 //one more user tested, bring the next
one
           begin
              busy <= 1'b1;
              enablew <= 1'b1;
              enabled <= 1'b1;
              writee <= 0;  //read
              address <= address + 1;
              count <= count + 1;
              intuser_count <= intuser_count - 1;
              next_state <= PROCESSING1;
                                  feat_comp0 <= data_out0;
```

```
                    feat_comp1 <= data_out1;
                    feat_comp2 <= data_out2;
                    feat_comp3 <= data_out3;
                    feat_comp4 <= data_out4;
                    feat_comp5 <= data_out5;
                    feat_comp6 <= data_out6;
                    feat_comp7 <= data_out7;
                    feat_comp8 <= data_out8;
                    feat_comp9 <= data_out9;
                    feat_comp10 <= data_out10;
                    feat_comp11 <= data_out11;
                    feat_comp12 <= data_out12;
                    feat_comp13 <= data_out13;
                    feat_comp14 <= data_out14;
                    feat_comp15 <= data_out15;
end
else //count != Total // haven't completed testing a user
begin
   busy <= 1'b1;
   enablew <= 1'b1;
   enabled <= 1'b1;
   writee <= 0;  //read
   address <= address + 1;
   count <= count + 1;
   intuser_count <= intuser_count;
   next_state <= PROCESSING1;
                    feat_comp0 <= data_out0;
                    feat_comp1 <= data_out1;
                    feat_comp2 <= data_out2;
                    feat_comp3 <= data_out3;
                    feat_comp4 <= data_out4;
                    feat_comp5 <= data_out5;
                    feat_comp6 <= data_out6;
                    feat_comp7 <= data_out7;
                    feat_comp8 <= data_out8;
                    feat_comp9 <= data_out9;
                    feat_comp10 <= data_out10;
                    feat_comp11 <= data_out11;
                    feat_comp12 <= data_out12;
                    feat_comp13 <= data_out13;
                    feat_comp14 <= data_out14;
                    feat_comp15 <= data_out15;
end
else //new data hasn't arrived
begin
   busy <= 1'b1;
```

```
            enablew <= 1'b1;
            enabled <= 0;
            writee <= 0;  //read
            address <= address;
            count <= count;
            intuser_count <= intuser_count;
            next_state <= PROCESSING1;
                              feat_comp0 <= data_out0;
                              feat_comp1 <= data_out1;
                              feat_comp2 <= data_out2;
                              feat_comp3 <= data_out3;
                              feat_comp4 <= data_out4;
                              feat_comp5 <= data_out5;
                              feat_comp6 <= data_out6;
                              feat_comp7 <= data_out7;
                              feat_comp8 <= data_out8;
                              feat_comp9 <= data_out9;
                              feat_comp10 <= data_out10;
                              feat_comp11 <= data_out11;
                              feat_comp12 <= data_out12;
                              feat_comp13 <= data_out13;
                              feat_comp14 <= data_out14;
                              feat_comp15 <= data_out15;
        end
    end


            default: begin
                                  if (allusers_used)
                                  begin
                                  intuser_count <= 2'b11;
                                  dis_count <= 2'b11;
                                  end
                                  else begin
                                  intuser_count <= user;
                                  dis_count <= user;
                                  end
                                  end
            endcase
            end
            endmodule
```

Validation:
By Raiza Muñiz

```
module validation(reset, clock, start, validated_user, validation_done,
dis_count, user0, user1, user2, user3);

input reset, clock, start;
input [1:0] dis_count;
input [47:0] user0, user1, user2, user3;
output validation_done;
output [3:0] validated_user;

reg busy, validation_done;
reg [1:0] state, next_state;
reg [3:0] validated_user;
reg [47:0] p_user0, p_user1, p_user2, p_user3;

parameter IDLE = 0;
parameter PROCESSING = 1;

always @(posedge clock)
if (reset)
   state <= IDLE;
else
state <= next_state;

always @ (state or start)
begin
   case (state)
      IDLE: begin
         if (~reset)
         if (start)
         begin
          if (dis_count == 2'b11)
          begin
           p_user0 <= user0;
             p_user1 <= user1;
             p_user2 <= user2;
             p_user3 <= user3;
             validation_done <= 0;
             next_state <= PROCESSING;
             end
            else
            if (dis_count == 2'b10)
            begin
            p_user0 <= user0;
             p_user1 <= user1;
             p_user2 <= user2;
             p_user3 <= 48'hffff_ffff_ffff;
```

```
        validation_done <= 0;
         next_state <= PROCESSING;
         end
         else if (dis_count == 2'b01)
         begin
         p_user0 <= user0;
         p_user1 <= user1;
         p_user2 <= 48'hffff_ffff_ffff;
         p_user3 <= 48'hffff_ffff_ffff;
         validation_done <= 0;
         next_state <= PROCESSING;
         end
        else // dis_count == 2'b00
        begin
         p_user0 <= user0;
         p_user1 <= 48'hffff_ffff_ffff;
         p_user2 <= 48'hffff_ffff_ffff;
         p_user3 <= 48'hffff_ffff_ffff;
        validation_done <= 0;
         next_state <= PROCESSING;
          end
          end
    else  // no start
    begin
         p_user0 <= 48'hffff_ffff_ffff;
         p_user1 <= 48'hffff_ffff_ffff;
         p_user2 <= 48'hffff_ffff_ffff;
         p_user3 <= 48'hffff_ffff_ffff;
         validation_done <= 0;
         next_state <= IDLE;
    end
      else // reset
      begin
         p_user0 <= 48'hffff_ffff_ffff;
         p_user1 <= 48'hffff_ffff_ffff;
         p_user2 <= 48'hffff_ffff_ffff;
         p_user3 <= 48'hffff_ffff_ffff;
         validation_done <= 0;
         next_state <= IDLE;
         end
        end

PROCESSING: begin
         validation_done <= 1'b1;
         next_state <= IDLE;
         validated_user <= (p_user0 < p_user1) ? (p_user0 < p_user2) ? (p_user0 <
```

```
p_user3) ? 4'b0000 : 4'b0011 : (p_user2 < p_user3) ? 4'b0010 : 4'b0011 : (p_user1 <
p_user2) ? (p_user1 < p_user3) ? 4'b0001 : 4'b0011: (p_user2 < p_user3) ? 4'b0010 :
4'b0011;
          end
   endcase
      end
endmodule
```

ACTION_REG:
By Raiza Muñiz

```
module action_reg(clock, reset, reg_reset, add, id, action);

input clock, reset, reg_reset, add, id;
output [1:0] action;

reg [1:0] action;
parameter IDLE = 0;
parameter ADD = 1;
parameter ID = 2;

always @(posedge clock)
if (reset | reg_reset)
action <= IDLE;
 else if (add)
  action <= ADD;
  else if (id)
  action <= ID;
  else action <= IDLE;

  endmodule
```

Delay:
By Raiza Muñiz

```
module delay(pixel_clock, hsync, vsync, vga_out_hsync, vga_out_vsync);
input pixel_clock, hsync, vsync;
output vga_out_hsync, vga_out_vsync;

reg vga_out_hsync, hsync_int, vga_out_vsync, vsync_int;

always @ (posedge pixel_clock)
begin
hsync_int <= hsync;
vga_out_hsync <= hsync_int;
vsync_int <= vsync;
```

```
vga_out_vsync <= vsync_int;
end  // endalways
endmodule
```

VGA:
by Raiza Muñiz

```
module vga(pixel_clock, reset, hsync, vsync, hblank, vblank, pixel_count,
line_count);

input pixel_clock;
input reset;
output hsync, vsync;
output hblank, vblank;
output [9:0] pixel_count;
output [9:0] line_count;

reg hsync, vsync;
reg hblank, vblank;
reg [9:0] pixel_count = 10'b0;
reg [9:0] line_count = 10'b0;
wire pixel_clock;
wire [9:0] next_pixel_count;
wire [9:0] next_line_count;

parameter PIXELS = 800;
parameter LINES = 525;
parameter HACTIVE_PIXELS = 640;
parameter HFRONT_PORCH = 16;
parameter HSYNC_PULSE = 96;
parameter HBACK_PORCH = 48;
parameter VACTIVE_LINES = 480;
parameter VFRONT_PORCH = 11;
parameter VSYNC_PULSE = 2;
parameter VBACK_PORCH = 32;

always @ (posedge pixel_clock)
begin
if (reset) // rest values counts and set syncs and blanks to high
begin
hsync <= 1'b1;
vsync <= 1'b1;
hblank <= 1'b1;
vblank <= 1'b1;
pixel_count <= 10'b0;
line_count <= 10'b0;
```

```
end

else
begin
hsync <= (next_pixel_count < HACTIVE_PIXELS + HFRONT_PORCH) |
(next_pixel_count
>=
HACTIVE_PIXELS + HFRONT_PORCH + HSYNC_PULSE);
vsync <= (next_line_count < VACTIVE_LINES + VFRONT_PORCH) |
(next_line_count >=
VACTIVE_LINES + VFRONT_PORCH + VSYNC_PULSE);
hblank <= (next_pixel_count < HACTIVE_PIXELS);
vblank <= (next_line_count < VACTIVE_LINES);
pixel_count <= next_pixel_count;
line_count <= next_line_count;
end
end

assign next_pixel_count = (pixel_count == PIXELS-1) ? 10'b0 : pixel_count +
1'b1;
assign next_line_count =  (pixel_count == PIXELS-1) ? (line_count == LINES-1) ?
10'b0 :
line_count + 1'b1 : line_count;


endmodule
```

Display:
By Raiza Muñiz

```
module displayfield(pixel_count, line_count, vgaoutput, red, green, blue);

input [9:0] pixel_count;
input [8:0] line_count;
input [3:0] vgaoutput;

output [7:0] red;
output [7:0] green;
output [7:0] blue;

parameter Dred = 24'b1111_1111_0000_1111_0000_1111;
parameter Dblue = 24'b0000_1111_0100_1111_1111_1111;
parameter Dgreen = 24'b0000_0011_1011_1111_0000_0101;
parameter black = 24'h0;
parameter white = 24'hffffff;
```

```
//wire [7:0] red, green, blue;

wire [23:0] color;
wire [23:0] color_r1, color_r2, color_r3, color_r4, color_r5, color_r6,
color_r7, color_r8, color_r9;
wire [23:0] color_e1, color_e2, color_e3, color_e4;
wire [23:0] color_a1, color_a2, color_a3, color_a4, color_a5, color_a6,
color_a7,
color_a8, color_a9, color_a10, color_a11, color_a12;
wire [23:0] color_d1, color_d2, color_d3, color_d4;
wire [23:0] color_y1, color_y2, color_y3, color_y4, color_y5, color_y6,
color_y7,
color_y8, color_y9, color_y10, color_y11;
wire [23:0] color_i1, color_i2, color_i3;
wire [23:0] color_u1, color_u2, color_u3;
wire [23:0] color_in1, color_in2;
wire [23:0] color_n11;
wire [23:0] color_n21, color_n22, color_n23, color_n24;
wire [23:0] color_n31, color_n32, color_n33, color_n34, color_n35;
wire [23:0] color_n41, color_n42, color_n43;


          //draw R
          rectangle letter_r1(pixel_count, line_count, 10'd30, 9'd105,
color_r1);
          defparam letter_r1.WIDTH    = 25;
          defparam letter_r1.HEIGHT  = 250;
          defparam letter_r1.COLOR = Dgreen;

          rectangle letter_r2(pixel_count, line_count, 10'd55, 9'd105,
color_r2);
          defparam letter_r2.WIDTH    = 85;
          defparam letter_r2.HEIGHT  = 25;
          defparam letter_r2.COLOR = Dgreen;

rectangle letter_r3(pixel_count, line_count, 10'd115, 9'd130, color_r3);
          defparam letter_r3.WIDTH    = 25;
          defparam letter_r3.HEIGHT  = 75;
          defparam letter_r3.COLOR = Dgreen;

         rectangle letter_r4(pixel_count, line_count, 10'd55, 9'd205,
color_r4);
          defparam letter_r4.WIDTH    = 85;
          defparam letter_r4.HEIGHT  = 25;
```

```
        defparam letter_r4.COLOR = Dgreen;

rectangle letter_r5(pixel_count, line_count, 10'd58, 9'd230, color_r5);
        defparam letter_r5.WIDTH    = 30;
        defparam letter_r5.HEIGHT  = 25;
        defparam letter_r5.COLOR = Dgreen;

rectangle letter_r6(pixel_count, line_count, 10'd72, 9'd255, color_r6);
        defparam letter_r6.WIDTH    = 34;
        defparam letter_r6.HEIGHT  = 25;
        defparam letter_r6.COLOR = Dgreen;

rectangle letter_r7(pixel_count, line_count, 10'd89, 9'd280, color_r7);
        defparam letter_r7.WIDTH= 34;
        defparam letter_r7.HEIGHT = 25;
        defparam letter_r7.COLOR = Dgreen;

rectangle letter_r8(pixel_count, line_count, 10'd106, 9'd305, color_r8);
        defparam letter_r8.WIDTH    = 34;
        defparam letter_r8.HEIGHT  = 25;
        defparam letter_r8.COLOR = Dgreen;

rectangle letter_r9(pixel_count, line_count, 10'd115, 9'd330, color_r9);
        defparam letter_r9.WIDTH    = 26;
        defparam letter_r9.HEIGHT  = 25;
        defparam letter_r9.COLOR = Dgreen;

//draw E
rectangle letter_e1(pixel_count, line_count, 10'd150, 9'd105, color_e1);
        defparam letter_e1.WIDTH   = 25;
        defparam letter_e1.HEIGHT  = 250;
        defparam letter_e1.COLOR = Dred;

rectangle letter_e2(pixel_count, line_count, 10'd175, 9'd105, color_e2);
        defparam letter_e2.WIDTH   = 85;
        defparam letter_e2.HEIGHT  = 25;
        defparam letter_e2.COLOR = Dred;

rectangle letter_e3(pixel_count, line_count, 10'd175, 9'd186, color_e3);
        defparam letter_e3.WIDTH   = 65;
        defparam letter_e3.HEIGHT  = 25;
        defparam letter_e3.COLOR = Dred;

rectangle letter_e4(pixel_count, line_count, 10'd175, 9'd330, color_e4);
        defparam letter_e4.WIDTH   = 85;
        defparam letter_e4.HEIGHT  = 25;
```

```
        defparam letter_e4.COLOR = Dred;


       //draw A
        rectangle letter_a1 (pixel_count, line_count, 10'd270, 9'd305,
color_a1);
        defparam letter_a1.WIDTH   = 15;
        defparam letter_a1.HEIGHT  = 50;
        defparam letter_a1.COLOR = Dred;

rectangle letter_a2 (pixel_count, line_count, 10'd280, 9'd255, color_a2);
        defparam letter_a2.WIDTH   = 15;
        defparam letter_a2.HEIGHT  = 60;
        defparam letter_a2.COLOR = Dred;

       rectangle letter_a3 (pixel_count, line_count, 10'd290, 9'd205,
color_a3);
        defparam letter_a3.WIDTH   = 15;
        defparam letter_a3.HEIGHT  = 60;
        defparam letter_a3.COLOR = Dred;

       rectangle letter_a4 (pixel_count, line_count, 10'd300, 9'd155,
color_a4);
        defparam letter_a4.WIDTH   = 15;
        defparam letter_a4.HEIGHT  = 60;
        defparam letter_a4.COLOR = Dred;

rectangle letter_a5 (pixel_count, line_count, 10'd310, 9'd105, color_a5);
        defparam letter_a5.WIDTH   = 10;
        defparam letter_a5.HEIGHT  = 60;
        defparam letter_a5.COLOR = Dred;

rectangle letter_a6 (pixel_count, line_count, 10'd320, 9'd105, color_a6);
        defparam letter_a6.WIDTH   = 15;
        defparam letter_a6.HEIGHT  = 30;
        defparam letter_a6.COLOR = Dred;

rectangle letter_a7 (pixel_count, line_count, 10'd330, 9'd105, color_a7);
        defparam letter_a7.WIDTH   = 15;
        defparam letter_a7.HEIGHT  = 60;
        defparam letter_a7.COLOR = Dred;

rectangle letter_a8 (pixel_count, line_count, 10'd340, 9'd155, color_a8);
        defparam letter_a8.WIDTH   = 15;
        defparam letter_a8.HEIGHT  = 60;
        defparam letter_a8.COLOR = Dred;
```

```
rectangle letter_a9 (pixel_count, line_count, 10'd350, 9'd205, color_a9);
        defparam letter_a9.WIDTH   = 15;
        defparam letter_a9.HEIGHT  = 60;
        defparam letter_a9.COLOR = Dred;

rectangle letter_a10 (pixel_count, line_count, 10'd360, 9'd255, color_a10);
        defparam letter_a10.WIDTH = 15;
        defparam letter_a10.HEIGHT = 60;
        defparam letter_a10.COLOR = Dred;

rectangle letter_a11 (pixel_count, line_count, 10'd370, 9'd305, color_a11);
        defparam letter_a11.WIDTH = 15;
        defparam letter_a11.HEIGHT = 50;
        defparam letter_a11.COLOR = Dred;

rectangle letter_a12 (pixel_count, line_count, 10'd300, 9'd209, color_a12);
        defparam letter_a12.WIDTH = 50;
        defparam letter_a12.HEIGHT = 30;
        defparam letter_a12.COLOR = Dred;

//draw d
rectangle letter_d1(pixel_count, line_count, 10'd390, 9'd105, color_d1);
        defparam letter_d1.WIDTH   = 25;
        defparam letter_d1.HEIGHT  = 250;
        defparam letter_d1.COLOR = Dblue;

rectangle letter_d2(pixel_count, line_count, 10'd415, 9'd105, color_d2);
        defparam letter_d2.WIDTH   = 60;
        defparam letter_d2.HEIGHT  = 25;
        defparam letter_d2.COLOR = Dblue;

rectangle letter_d3(pixel_count, line_count, 10'd455, 9'd130, color_d3);
        defparam letter_d3.WIDTH   = 35;
        defparam letter_d3.HEIGHT  = 200; ///
        defparam letter_d3.COLOR = Dblue;

rectangle letter_d4(pixel_count, line_count, 10'd415, 9'd330, color_d4);
        defparam letter_d4.WIDTH   = 60;
        defparam letter_d4.HEIGHT  = 25;
        defparam letter_d4.COLOR = Dblue;

//draw y
rectangle letter_y1(pixel_count, line_count, 10'd560, 9'd224, color_y1);
        defparam letter_y1.WIDTH   = 20;
        defparam letter_y1.HEIGHT  = 125;
```

```
        defparam letter_y1.COLOR = Dblue;

rectangle letter_y2(pixel_count, line_count, 10'd554, 9'd205, color_y2);
        defparam letter_y2.WIDTH   = 16;
        defparam letter_y2.HEIGHT  = 25;
        defparam letter_y2.COLOR = Dblue;

rectangle letter_y3(pixel_count, line_count, 10'd543, 9'd180, color_y3);
        defparam letter_y3.WIDTH   = 16;
        defparam letter_y3.HEIGHT  = 25;
        defparam letter_y3.COLOR = Dblue;

rectangle letter_y4(pixel_count, line_count, 10'd532, 9'd155, color_y4);
        defparam letter_y4.WIDTH   = 16;
        defparam letter_y4.HEIGHT  = 25;
        defparam letter_y4.COLOR = Dblue;

rectangle letter_y5(pixel_count, line_count, 10'd521, 9'd130, color_y5);
        defparam letter_y5.WIDTH   = 16;
        defparam letter_y5.HEIGHT  = 25;
        defparam letter_y5.COLOR = Dblue;

rectangle letter_y6(pixel_count, line_count, 10'd510, 9'd105, color_y6);
        defparam letter_y6.WIDTH   = 16;
        defparam letter_y6.HEIGHT  = 25;
        defparam letter_y6.COLOR = Dblue;

rectangle letter_y7(pixel_count, line_count, 10'd574, 9'd205, color_y7);
        defparam letter_y7.WIDTH   = 16;
        defparam letter_y7.HEIGHT  = 25;
        defparam letter_y7.COLOR = Dblue;

rectangle letter_y8(pixel_count, line_count, 10'd584, 9'd180, color_y8);
        defparam letter_y8.WIDTH   = 16;
        defparam letter_y8.HEIGHT  = 25;
        defparam letter_y8.COLOR = Dblue;

rectangle letter_y9(pixel_count, line_count, 10'd594, 9'd155, color_y9);
        defparam letter_y9.WIDTH   = 16;
        defparam letter_y9.HEIGHT  = 25;
        defparam letter_y9.COLOR = Dblue;

rectangle letter_y10(pixel_count, line_count, 10'd604, 9'd130, color_y10);
        defparam letter_y10.WIDTH = 16;
        defparam letter_y10.HEIGHT          = 25;
        defparam letter_y10.COLOR = Dblue;
```

```
rectangle letter_y11(pixel_count, line_count, 10'd614, 9'd105, color_y11);
        defparam letter_y11.WIDTH = 16;
        defparam letter_y11.HEIGHT          = 25;
        defparam letter_y11.COLOR = Dblue;


        // draw I
        rectangle letter_i1(pixel, line, 10'd190, 9'd100, color_i1);
        defparam letter_i1.WIDTH    = 120;
        defparam letter_i1.HEIGHT  = 39;
        defparam letter_i1.COLOR  = Dred;

        rectangle letter_i2(pixel, line, 10'd388, 9'd200, color_i2);
        defparam letter_i2.WIDTH    = 120;
        defparam letter_i2.HEIGHT  = 39;
        defparam letter_i2.COLOR                = Dred;

       rectangle letter_i3(pixel, line, 10'd289, 9'd100, color_i3);
        defparam letter_i3.WIDTH   = 30;
        defparam letter_i3.HEIGHT  = 61;
        defparam letter_i3.COLOR                = Dred;

       //draw u
       rectangle letter_u1(pixel, line, 10'd190, 9'd100, color_u1);
        defparam letter_u1.WIDTH   = 39;
        defparam letter_u1.HEIGHT  = 210;
        defparam letter_u1.COLOR                = white;

       rectangle letter_u2(pixel, line, 10'd331, 9'd100, color_u2);
        defparam letter_u2.WIDTH   = 39;
        defparam letter_u2.HEIGHT  = 210;
        defparam letter_u2.COLOR                = white;

       rectangle letter_u3(pixel, line, 10'd230, 9'd271, color_u3);
        defparam letter_u3.WIDTH   = 100;
        defparam letter_u3.HEIGHT  = 39;
        defparam letter_u3.COLOR                = white;

       //draw in
       rectangle letter_in1(pixel, line, 10'd390, 9'd100, color_in1);
        defparam letter_in1.WIDTH  = 39;
        defparam letter_in1.HEIGHT= 39;
        defparam letter_in1.COLOR               = Dred;

       rectangle letter_in2(pixel, line, 10'd390, 9'd169, color_in2);
```

```
 defparam letter_in2.WIDTH  = 39;
 defparam letter_in2.HEIGHT= 41;
 defparam letter_in2.COLOR                = Dred;


//draw 1
rectangle letter_n11(pixel, line, 10'd390, 9'd100, color_n11);
 defparam letter_n11.WIDTH = 39;
 defparam letter_n11.HEIGHT= 210;
 defparam letter_n11.COLOR                = Dblue;

//draw 2
rectangle letter_n21(pixel, line, 10'd390, 9'd100, color_n21);
 defparam letter_n21.WIDTH = 100;
 defparam letter_n21.HEIGHT= 39;
 defparam letter_n21.COLOR                = Dblue;

rectangle letter_n22(pixel, line, 10'd451, 9'd100, color_n22);
 defparam letter_n22.WIDTH = 39;
 defparam letter_n22.HEIGHT= 66;
 defparam letter_n22.COLOR                = Dblue;

rectangle letter_n23(pixel, line, 10'd390, 9'd167, color_n23);
 defparam letter_n23.WIDTH = 100;
 defparam letter_n23.HEIGHT= 39;
 defparam letter_n23.COLOR                = Dblue;

rectangle letter_n24(pixel, line, 10'd390, 9'd205, color_n24);
 defparam letter_n24.WIDTH = 100;
 defparam letter_n24.HEIGHT= 39;
 defparam letter_n24.COLOR                = Dblue;

// draw 3
rectangle letter_n31(pixel, line, 10'd390, 9'd100, color_n31);
 defparam letter_n31.WIDTH = 100;
 defparam letter_n31.HEIGHT= 39;
 defparam letter_n31.COLOR                = Dblue;

rectangle letter_n32(pixel, line, 10'd451, 9'd140, color_n32);
 defparam letter_n32.WIDTH = 39;
 defparam letter_n32.HEIGHT= 8;
 defparam letter_n32.COLOR                = Dblue;

rectangle letter_n33(pixel, line, 10'd390, 9'd149, color_n33);
 defparam letter_n33.WIDTH = 100;
 defparam letter_n33.HEIGHT= 16;
```

```
            defparam letter_n33.COLOR               = Dblue;

           rectangle letter_n34(pixel, line, 10'd451, 9'd166, color_n34);
            defparam letter_n34.WIDTH = 39;
            defparam letter_n34.HEIGHT= 8;
            defparam letter_n34.COLOR               = Dblue;

           rectangle letter_n35(pixel, line, 10'd390, 9'd171, color_n35);
            defparam letter_n35.WIDTH = 100;
            defparam letter_n35.HEIGHT= 39;
            defparam letter_n35.COLOR               = Dblue;

           //draw 4
rectangle letter_n41(pixel, line, 10'd390, 9'd100, color_n41);
            defparam letter_n41.WIDTH = 100;
            defparam letter_n41.HEIGHT= 39;
            defparam letter_n41.COLOR               = Dblue;

           rectangle letter_n42(pixel, line, 10'd451, 9'd140, color_n42);
            defparam letter_n42.WIDTH = 39;
            defparam letter_n42.HEIGHT= 210;
            defparam letter_n42.COLOR               = Dblue;

           rectangle letter_n43(pixel, line, 10'd390, 9'd100, color_43);
            defparam letter_n43.WIDTH = 100;
            defparam letter_n43.HEIGHT= 39;
            defparam letter_n43.COLOR               = Dblue;



assign color = (vgaoutput == 4'b0000) ? (color_u1 | color_u2 | color_u3 |
color_in1 |
color_in2 | color_n11) : (vgaoutput == 4'b0001) ?  (color_u1 | color_u2 |
color_u3
|color_n21 | color_n22 | color_n23 | color_n24): (vgaoutput == 4'b0010) ?
(color_u1 | color_u2 | color_u3 | color_n31 | color_n32 | color_n33| color_n34 |
color_n35) : (vgaoutput == 4'b0011) ? (color_u1 | color_u2 | color_u3 |
color_n41 |
color_n42 | color_n43) : (vgaoutput == 4'b1100) ? (color_i1 | color_i2 |
color_i3) : (vgaoutput == 4'b1000) ? (color_e1 | color_e2 | color_e3 | color_e4)
:
(vgaoutput == 4'b1001) ? (color_e1 | color_e2 | color_e3 | color_e4 | color_d1 |
color_d2 |
color_d3 | color_d4) : (vgaoutput == 4'b1010) ? (color_a1 | color_a2 | color_a3
| color_a4 | color_a5 | color_a6 | color_a7 |
color_a8 | color_a9 | color_a10 | color_a11| color_a12 | color_d1 | color_d2 |
```

```
color_d3 | color_d4) : (vgaoutput == 4'b1011) ?
(color_r1 | color_r2 | color_r3 | color_r4 | color_r5 | color_r6 | color_r7 |
color_r8 |
color_r9 | color_e1 | color_e2 | color_e3 | color_e4 | color_a1 | color_a2 |
color_a3 | color_a4 | color_a5 | color_a6 | color_a7 |
color_a8 | color_a9 | color_a10 | color_a11| color_a12 | color_d1 | color_d2 |
color_d3 | color_d4 | color_y1 | color_y2 | color_y3 | color_y4 | color_y5 |
color_y6 | color_y7 |
color_y8 | color_y9 | color_y10 | color_y11) : (vgaoutput == 4'b1101) ?
(color_a1 | color_a2 | color_a3 | color_a4 |
color_a5 | color_a6 | color_a7 | color_a8 | color_a9 | color_a10 | color_a11 |
color_a12) :
(color_in1 | color_in2);

assign red = color[23:16];
assign green = color[15:8];
assign blue = color[7:0];


endmodule
```

- Raiza

```
// The labkit code changes for Jaime's implementations was:

wire reset;
debounce deb_reset(~button0,pixel_clock,~button0,reset);

wire start;
debounce deb_start(~button0,pixel_clock,~button1,start);

//wire new_current, new_stored, endd;
//wire [15:0] cf_1, cf_2, cf_3, cf_4, cf_5, cf_6, cf_7, cf_8, cf_9, cf_10, cf_11, cf_12,
cf_13, cf_14, cf_15, cf_16;
//
//distance_testing dt(pixel_clock,reset,start,new_current,new_stored, endd, cf_1, cf_2,
cf_3, cf_4,
//            cf_5, cf_6, cf_7, cf_8, cf_9, cf_10, cf_11, cf_12, cf_13, cf_14, cf_15, cf_16);
//
//wire [15:0] pf_1, pf_2, pf_3, pf_4, pf_5, pf_6, pf_7, pf_8, pf_9, pf_10, pf_11, pf_12,
pf_13, pf_14, pf_15, pf_16;
//
//assign pf_1 = 16'hFE2A;
//assign pf_2 = 16'hC823;
```

```
//assign pf_3 = 16'h2B11;
//assign pf_4 = 16'h5486;
//assign pf_5 = 16'h445D;
//assign pf_6 = 16'h11D1;
//assign pf_7 = 16'h67CA;
//assign pf_8 = 16'h891A;
//assign pf_9 = 16'hBA44;
//assign pf_10 = 16'hE114;
//assign pf_11 = 16'h1FF3;
//assign pf_12 = 16'hCF89;
//assign pf_13 = 16'hAB00;
//assign pf_14 = 16'h1111;
//assign pf_15 = 16'h1313;
//assign pf_16 = 16'h0022;
//
//wire [1:0] user;
//assign user = 2'b11;
//wire [63:0] user1_out, user2_out, user3_out, user4_out;
//wire ready;
//
//distance_processor dp(pixel_clock, reset, new_stored, new_current, endd, user,
//                      user1_out, user2_out, user3_out, user4_out, ready, cf_1,
cf_2, cf_3, cf_4,
//                      cf_5, cf_6, cf_7, cf_8, cf_9, cf_10, cf_11, cf_12, cf_13,
cf_14, cf_15, cf_16,
//                      pf_1, pf_2, pf_3, pf_4, pf_5, pf_6, pf_7, pf_8, pf_9, pf_10,
pf_11, pf_12,
//                      pf_13, pf_14, pf_15, pf_16);


wire sdata_out, sync, audio_reset_b, beep, fft_start;
wire signed [17:0] audio;

wire mem_wr_clk;
wire signed [17:0] audio_temp;

ac_97_controller ac97c(ac97_bit_clock, reset, ac97_sdata_in, ac97_sdata_out,
                                                              ac97_synch,
audio_reset_b, beep, audio_temp, mem_wr_clk);

wire read, write, full, empty;

audio_mem_control a_m_c(pixel_clock,reset,start,read,write,full,empty,fft_start);

audio_temp_storage
a_t_s(.wr_clk(mem_wr_clk),.rd_clk(pixel_clock),.full(full),.empty(empty),
```

```
                .rd_en(read),.wr_en(write),.rst(reset),.din(audio_temp),.dout(audio));

wire signed [17:0] x_real, xn_re, x_im, xn_im, xk_re, xk_im;
assign xn_im = 18'b0;

wire fwd_inv, fwd_inv_we, done, edone, dv, busy, rfd;
wire [10:0] xk_index, xn_index;
assign fwd_inv_we = 0;
assign fwd_inv = 1;
//assign xn_re = 18'h008FF;
//
//wire fft_s;
//assign fft_s = 1;
//
fft1 fastfouriertrans(.xn_re(audio),.xn_im(xn_im),.clk(pixel_clock),.start(fft_start),
      .xk_re(xk_re), .xk_im(xk_im), .fwd_inv(fwd_inv), .fwd_inv_we(fwd_inv_we),
.xk_index(xk_index), .xn_index(xn_index),
                        .rfd(rfd), .busy(busy), .dv(dv), .edone(edone), .done(done));

wire [10:0] index_out_1;
wire done_1;
wire signed [36:0] x_power;
power_convertor
pc(pixel_clock,reset,xk_re,xk_im,xn_index,done,index_out_1,x_power,done_1);

wire [10:0]
index_1,index_2,index_3,index_4,index_5,index_6,index_7,index_8,index_9,index_10;
wire [10:0]
index_11,index_12,index_13,index_14,index_15,index_16,index_17,index_18,index_19;
wire [35:0] filt_1,filt_2,filt_3,filt_4,filt_5,filt_6,filt_7,filt_8,filt_9,filt_10;
wire [35:0] filt_11,filt_12,filt_13,filt_14,filt_15,filt_16,filt_17,filt_18,filt_19;
wire
done_2_1,done_2_2,done_2_3,done_2_4,done_2_5,done_2_6,done_2_7,done_2_8,done_
2_9,done_2_10;
wire
done_2_11,done_2_12,done_2_13,done_2_14,done_2_15,done_2_16,done_2_17,done_2
_18,done_2_19;

filter1 f1(clock, reset, done_1, index_out_1, x_power, index_1, filt_1, done_2_1);
filter2 f2(clock, reset, done_1, index_out_1, x_power, index_2, filt_2, done_2_2);
filter3 f3(clock, reset, done_1, index_out_1, x_power, index_3, filt_3, done_2_3);
filter4 f4(clock, reset, done_1, index_out_1, x_power, index_4, filt_4, done_2_4);
filter5 f5(clock, reset, done_1, index_out_1, x_power, index_5, filt_5, done_2_5);
filter6 f6(clock, reset, done_1, index_out_1, x_power, index_6, filt_6, done_2_6);
filter7 f7(clock, reset, done_1, index_out_1, x_power, index_7, filt_7, done_2_7);
filter8 f8(clock, reset, done_1, index_out_1, x_power, index_8, filt_8, done_2_8);
```

```
filter9 f9(clock, reset, done_1, index_out_1, x_power, index_9, filt_9, done_2_9);
filter10 f10(clock, reset, done_1, index_out_1, x_power, index_10, filt_10, done_2_10);
filter11 f11(clock, reset, done_1, index_out_1, x_power, index_11, filt_11, done_2_11);
filter12 f12(clock, reset, done_1, index_out_1, x_power, index_12, filt_12, done_2_12);
filter13 f13(clock, reset, done_1, index_out_1, x_power, index_13, filt_13, done_2_13);
filter14 f14(clock, reset, done_1, index_out_1, x_power, index_14, filt_14, done_2_14);
filter15 f15(clock, reset, done_1, index_out_1, x_power, index_15, filt_15, done_2_15);
filter16 f16(clock, reset, done_1, index_out_1, x_power, index_16, filt_16, done_2_16);
filter17 f17(clock, reset, done_1, index_out_1, x_power, index_17, filt_17, done_2_17);
filter18 f18(clock, reset, done_1, index_out_1, x_power, index_18, filt_18, done_2_18);
filter19 f19(clock, reset, done_1, index_out_1, x_power, index_19, filt_19, done_2_19);



// the actual code for the modules is:



module audio_front_end(pixel_clock,ac97_bit_clock,reset,start,sdata_in,sdata_out,sync,

audio_reset_b,beep,audio,fft_start);

input pixel_clock, ac97_bit_clock, reset, start, sdata_in;
output sdata_out, sync, audio_reset_b, beep, fft_start;
output [17:0] audio;

wire mem_wr_clk;
wire signed [17:0] audio_temp;

ac_97_controller ac97c(ac97_bit_clock, reset, sdata_in, sdata_out,
                                                                    sync, audio_reset_b,
beep, audio_temp, mem_wr_clk);

wire read, write, full, empty, fft_start;

audio_mem_control a_m_c(pixel_clock,reset,start,read,write,full,empty,fft_start);

audio_temp_storage
a_t_s(.wr_clk(mem_wr_clk),.rd_clk(pixel_clock),.full(full),.empty(empty),
                .rd_en(read),.wr_en(write),.rst(reset),.din(audio_temp),.dout(audio));


endmodule
```

```
module
ac_97_controller(clock,reset,sdata_in,sdata_out,sync,audio_reset_b,beep,audio_in,enable
);

input clock, reset, sdata_in;

output sdata_out, sync, audio_reset_b, beep, enable;
output [17:0] audio_in;

parameter ON = 1'b1;
parameter OFF = 1'b0;

wire beep,audio_reset_b;
assign audio_reset_b = ~reset;
assign beep = 1'b0;

reg state;
reg sdata_out;
reg sync;
reg [7:0] count;

always @(posedge clock) begin

  if (reset == ON) begin
                        state <= 1;
                        count <= 8'b0;
                        sync <= 1'b0;
                        sdata_out <= 1'b0;
                        end
            else begin
                        count <= count + 1;
                        if (count == 14) sync <= 1'b0;
```

```
                else if (count == 254) begin
                        sync <= 1'b1;
                        state <= state + 1;
                        end
                else sync <= sync;
                if (count == 255) sdata_out <= 1'b1;
                else if (count == 2) sdata_out <= 1'b0;
                case (state)

                        0 :     begin
                                        if (count == 21) sdata_out <= 1'b1;
                                        else if (count == 22) sdata_out <= 0'b0;
                                        end
                        1 :     begin
                                        if (count == 18) sdata_out <= 1'b1;
                                        else if (count == 21) sdata_out <= 1'b0;
                                        else if (count == 42) sdata_out <= 1'b1;
                                        else if (count == 43) sdata_out <= 1'b0;
                                        end

                endcase
                end
        end

reg [17:0] audio_in;
reg [19:0] temp_audio_in;
reg enable;

always @(posedge clock) begin
        if (reset == ON) begin
                temp_audio_in <= 20'b0;
                audio_in <= 16'b0;
                end
        else begin
                if ((count > 55) && (count < 76)) temp_audio_in <=
{temp_audio_in[18:0],sdata_in};
                else if (count == 76) begin
                        audio_in <= temp_audio_in[19:2];
                        enable <= 1'b1;
                        end
                else if (count == 203) enable <= 1'b0;
                else enable <= enable;
                end
        end

endmodule
```

```
module audio_mem_control(clock,reset,start,read,write,full,empty,fft_start);

input clock,reset,start,full,empty;
output read,write,fft_start;

parameter ON = 1;
parameter OFF = 0;
parameter IDLE = 0;
parameter RECORDING = 1;

reg read,write,fft_start;
reg state;
reg [7:0] count;

always @(posedge clock) begin

if (reset) begin
  state <= IDLE;
          count <= 0;
          write <= OFF;
          read <= OFF;
          fft_start <= OFF;
  end

else if (start) begin
  state <= RECORDING;
```

```
            count <= 0;
            write <= ON;
            read <= OFF;
            fft_start <= OFF;
            end

else if (state == RECORDING) begin

  if (count >= 160) begin
                  state <= IDLE;
                  write <= OFF;
                  read <= OFF;
                  fft_start <= OFF;
                  end
            else if ((full == ON) && (write == ON)) begin
              write <= OFF;
                  count <= count + 1;
                  read <= ON;
                  fft_start <= ON;
                  end
            else if (empty == ON) begin
              write <= ON;
                  count <= count;
                  read <= OFF;
                  fft_start <= OFF;
                  end
            else begin
                  read <= read;
                  count <= count;
                  fft_start <= OFF;
                  end


            end

else begin // IDLE state
            fft_start <= OFF;
            state <= IDLE;
            count <= 0;
            write <= OFF;
            read <= OFF;
            end

end

endmodule
```

```
module audio_back_end(clock,reset,x_real,x_im,index_in,start,time_int,cep1,cep2,cep3,

cep4,cep5,cep6,cep7,cep8,cep9,cep10,cep11,cep12,cep13,cep14,cep15,cep16,done);

input clock, reset, start;
input signed [17:0] x_real, x_im;
input [10:0] index_in;

output [9:0] time_int;
output done;
output signed [45:0]
cep1,cep2,cep3,cep4,cep5,cep6,cep7,cep8,cep9,cep10,cep11,cep12,cep13,cep14,cep15,ce
p16;

wire clock,reset,start,done;
wire signed [17:0] x_real, x_im;
wire [10:0] index_in;
wire [9:0] time_int;
wire signed [45:0]
cep1,cep2,cep3,cep4,cep5,cep6,cep7,cep8,cep9,cep10,cep11,cep12,cep13,cep14,cep15,ce
p16;
```

```
wire [10:0] index_out_1;
wire done_1;
wire signed [36:0] x_power;
power_convertor
pc(clock,reset,x_real,x_im,index_in,start,index_out_1,x_power,done_1);

wire [10:0]
index_1,index_2,index_3,index_4,index_5,index_6,index_7,index_8,index_9,index_10;
wire
[10:0]index_11,index_12,index_13,index_14,index_15,index_16,index_17,index_18,inde
x_19;
wire [35:0] filt_1,filt_2,filt_3,filt_4,filt_5,filt_6,filt_7,filt_8,filt_9,filt_10;
wire [35:0] filt_11,filt_12,filt_13,filt_14,filt_15,filt_16,filt_17,filt_18,filt_19;
wire
done_2_1,done_2_2,done_2_3,done_2_4,done_2_5,done_2_6,done_2_7,done_2_8,done_
2_9,done_2_10;
wire
done_2_11,done_2_12,done_2_13,done_2_14,done_2_15,done_2_16,done_2_17,done_2
_18,done_2_19;

filter1 f1(clock, reset, done_1, index_out_1, x_power, index_1, filt_1, done_2_1);
filter2 f2(clock, reset, done_1, index_out_1, x_power, index_2, filt_2, done_2_2);
filter3 f3(clock, reset, done_1, index_out_1, x_power, index_3, filt_3, done_2_3);
filter4 f4(clock, reset, done_1, index_out_1, x_power, index_4, filt_4, done_2_4);
filter5 f5(clock, reset, done_1, index_out_1, x_power, index_5, filt_5, done_2_5);
filter6 f6(clock, reset, done_1, index_out_1, x_power, index_6, filt_6, done_2_6);
filter7 f7(clock, reset, done_1, index_out_1, x_power, index_7, filt_7, done_2_7);
filter8 f8(clock, reset, done_1, index_out_1, x_power, index_8, filt_8, done_2_8);
filter9 f9(clock, reset, done_1, index_out_1, x_power, index_9, filt_9, done_2_9);
filter10 f10(clock, reset, done_1, index_out_1, x_power, index_10, filt_10, done_2_10);
filter11 f11(clock, reset, done_1, index_out_1, x_power, index_11, filt_11, done_2_11);
filter12 f12(clock, reset, done_1, index_out_1, x_power, index_12, filt_12, done_2_12);
filter13 f13(clock, reset, done_1, index_out_1, x_power, index_13, filt_13, done_2_13);
filter14 f14(clock, reset, done_1, index_out_1, x_power, index_14, filt_14, done_2_14);
filter15 f15(clock, reset, done_1, index_out_1, x_power, index_15, filt_15, done_2_15);
filter16 f16(clock, reset, done_1, index_out_1, x_power, index_16, filt_16, done_2_16);
filter17 f17(clock, reset, done_1, index_out_1, x_power, index_17, filt_17, done_2_17);
filter18 f18(clock, reset, done_1, index_out_1, x_power, index_18, filt_18, done_2_18);
filter19 f19(clock, reset, done_1, index_out_1, x_power, index_19, filt_19, done_2_19);


wire
ready_1,ready_2,ready_3,ready_4,ready_5,ready_6,ready_7,ready_8,ready_9,ready_10;
wire
ready_11,ready_12,ready_13,ready_14,ready_15,ready_16,ready_17,ready_18,ready_19;
```

```
wire [62:0] acc_1,acc_2,acc_3,acc_4,acc_5,acc_6,acc_7,acc_8,acc_9,acc_10;
wire [62:0] acc_11,acc_12,acc_13,acc_14,acc_15,acc_16,acc_17,acc_18,acc_19;
wire [9:0] time_1,time_2,time_3,time_4,time_5,time_6,time_7,time_8,time_9,time_10;
wire [9:0] time_11,time_12,time_13,time_14,time_15,time_16,time_17,time_18,time_19;

accumulator a(clock, reset, done_2_1, index_1, filt_1, acc_1, time_int, ready_1);
accumulator a1(clock, reset, done_2_2, index_2, filt_2, acc_2, time_2, ready_2);
accumulator a2(clock, reset, done_2_3, index_3, filt_3, acc_3, time_3, ready_3);
accumulator a3(clock, reset, done_2_4, index_4, filt_4, acc_4, time_4, ready_4);
accumulator a4(clock, reset, done_2_5, index_5, filt_5, acc_5, time_5, ready_5);
accumulator a5(clock, reset, done_2_6, index_6, filt_6, acc_6, time_6, ready_6);
accumulator a6(clock, reset, done_2_7, index_7, filt_7, acc_7, time_7, ready_7);
accumulator a7(clock, reset, done_2_8, index_8, filt_8, acc_8, time_8, ready_8);
accumulator a8(clock, reset, done_2_9, index_9, filt_9, acc_9, time_9, ready_9);
accumulator a9(clock, reset, done_2_10, index_10, filt_10, acc_10, time_10, ready_10);
accumulator a10(clock, reset, done_2_11, index_11, filt_11, acc_11, time_11, ready_11);
accumulator a11(clock, reset, done_2_12, index_12, filt_12, acc_12, time_12, ready_12);
accumulator a12(clock, reset, done_2_13, index_13, filt_13, acc_13, time_13, ready_13);
accumulator a13(clock, reset, done_2_14, index_14, filt_14, acc_14, time_14, ready_14);
accumulator a14(clock, reset, done_2_15, index_15, filt_15, acc_15, time_15, ready_15);
accumulator a15(clock, reset, done_2_16, index_16, filt_16, acc_16, time_16, ready_16);
accumulator a16(clock, reset, done_2_17, index_17, filt_17, acc_17, time_17, ready_17);
accumulator a17(clock, reset, done_2_18, index_18, filt_18, acc_18, time_18, ready_18);
accumulator a18(clock, reset, done_2_19, index_19, filt_19, acc_19, time_19, ready_19);

wire
computed_1,computed_2,computed_3,computed_4,computed_5,computed_6,computed_
7,computed_8,computed_9,computed_10;
wire
computed_11,computed_12,computed_13,computed_14,computed_15,computed_16,com
puted_17,computed_18,computed_19;
wire [16:0] log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,log2_9,log2_10;
wire [16:0]
log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,log2_17,log2_18,log2_19;

lg lg1(clock,reset,ready_1,acc_1,log2_1,computed_1);
lg lg2(clock,reset,ready_2,acc_2,log2_2,computed_2);
lg lg3(clock,reset,ready_3,acc_3,log2_3,computed_3);
lg lg4(clock,reset,ready_4,acc_4,log2_4,computed_4);
lg lg5(clock,reset,ready_5,acc_5,log2_5,computed_5);
lg lg6(clock,reset,ready_6,acc_6,log2_6,computed_6);
lg lg7(clock,reset,ready_7,acc_7,log2_7,computed_7);
lg lg8(clock,reset,ready_8,acc_8,log2_8,computed_8);
lg lg9(clock,reset,ready_9,acc_9,log2_9,computed_9);
lg lg10(clock,reset,ready_10,acc_10,log2_10,computed_10);
lg lg11(clock,reset,ready_11,acc_11,log2_11,computed_11);
```

```
lg lg12(clock,reset,ready_12,acc_12,log2_12,computed_12);
lg lg13(clock,reset,ready_13,acc_13,log2_13,computed_13);
lg lg14(clock,reset,ready_14,acc_14,log2_14,computed_14);
lg lg15(clock,reset,ready_15,acc_15,log2_15,computed_15);
lg lg16(clock,reset,ready_16,acc_16,log2_16,computed_16);
lg lg17(clock,reset,ready_17,acc_17,log2_17,computed_17);
lg lg18(clock,reset,ready_18,acc_18,log2_18,computed_18);
lg lg19(clock,reset,ready_19,acc_19,log2_19,computed_19);

wire done_2,done_3,done_4,done_5,done_6,done_7,done_8,done_9,done_10;
wire done_11,done_12,done_13,done_14,done_15,done_16,done_17,done_18,done_19;


dct1 dc1(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_1,cep1,done);
dct2 dc2(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_2,cep2,done_2);
dct1 dc3(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_3,cep3,done_3);
dct4 dc4(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_4,cep4,done_4);
dct1 dc5(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_5,cep5,done_5);
dct1 dc6(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_6,cep6,done_6);
dct1 dc7(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_7,cep7,done_7);
dct8 dc8(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_8,cep8,done_8);
dct1 dc9(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_9,cep9,done_9);
dct1 dc10(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_10,cep10,done_10);
dct1 dc11(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_11,cep11,done_11);
```

```
dct1 dc12(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_12,cep12,done_12);
dct1 dc13(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_13,cep13,done_13);
dct1 dc14(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_14,cep14,done_14);
dct1 dc15(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_15,cep15,done_15);
dct1 dc16(clock,reset,log2_1,log2_2,log2_3,log2_4,log2_5,log2_6,log2_7,log2_8,
        log2_9 ,log2_10,log2_11,log2_12,log2_13,log2_14,log2_15,log2_16,
        log2_17,log2_18,log2_19,computed_16,cep16,done_16);




endmodule
```

```
module
power_convertor(clock,reset,x_real,x_im,index_in,start,index_out,x_power,done);

input clock, reset, start;
input signed [17:0] x_real, x_im;
input [10:0] index_in;

output done;
output [10:0] index_out;
output signed [36:0] x_power;

reg [10:0] index_out;
reg signed [36:0] x_power;
```

```
reg done, state;
reg [11:0] count;

parameter NO = 0;
parameter YES = 1;
parameter UNLOADING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   x_power <= 0;
   state <= IDLE;
   count <= 0;
   done <= NO;
   end
else if (start) begin
   state <= UNLOADING;
   count <= 0;
   done <= YES;
   end
else if (state == UNLOADING) begin
   done <= NO;
   count <= count + 1;
   if (count == 2047) state <= IDLE;
   else state <= state;
   index_out <= index_in;
   x_power <= (x_real * x_real) + (x_im * x_im);
   end
else begin // state == IDLE
   index_out <= index_out;
   x_power <= x_power;
   count <= 0;
   done <= NO;
   end

end

endmodule
```

```
module filter1(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
```

```
  else state <= state;
  index_out <= index_in;
  case (index_in)
    1 : data_out <= ((data_in * $signed(8'd25)) >> 9);
    2 : data_out <= ((data_in * $signed(8'd50)) >> 9);
    3 : data_out <= ((data_in * $signed(8'd75)) >> 9);
    4 : data_out <= ((data_in * $signed(8'd99)) >> 9);
    5 : data_out <= ((data_in * $signed(8'd78)) >> 9);
    6 : data_out <= ((data_in * $signed(8'd56)) >> 9);
    7 : data_out <= ((data_in * $signed(8'd34)) >> 9);
    8 : data_out <= ((data_in * $signed(8'd12)) >> 9);
    default : data_out <= 36'b0;
  endcase
  end

else begin // IDLE state
  count <= 0;
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end


end

endmodule
```

```
module filter2(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;
```

```
reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    5 : data_out <= ((data_in * $signed(8'd19)) >> 9);
    6 : data_out <= ((data_in * $signed(8'd38)) >> 9);
    7 : data_out <= ((data_in * $signed(8'd58)) >> 9);
    8 : data_out <= ((data_in * $signed(8'd77)) >> 9);
    9 : data_out <= ((data_in * $signed(8'd80)) >> 9);
    10 : data_out <= ((data_in * $signed(8'd63)) >> 9);
    11 : data_out <= ((data_in * $signed(8'd46)) >> 9);
    12 : data_out <= ((data_in * $signed(8'd29)) >> 9);
    13 : data_out <= ((data_in * $signed(8'd12)) >> 9);
    default : data_out <= 36'b0;
  endcase
  end
```

```
else begin // IDLE state
   count <= 0;
   index_out <= index_out;
   data_out <= data_out;
   done <= NO;
   end


end

endmodule
```

```
module filter3(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   data_out <= 36'b0;
```

```
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    9 : data_out <= ((data_in * $signed(8'd7)) >> 9);
    10 : data_out <= ((data_in * $signed(8'd22)) >> 9);
    11 : data_out <= ((data_in * $signed(8'd37)) >> 9);
    12 : data_out <= ((data_in * $signed(8'd52)) >> 9);
    13 : data_out <= ((data_in * $signed(8'd67)) >> 9);
    14 : data_out <= ((data_in * $signed(8'd74)) >> 9);
    15 : data_out <= ((data_in * $signed(8'd61)) >> 9);
    16 : data_out <= ((data_in * $signed(8'd47)) >> 9);
    17 : data_out <= ((data_in * $signed(8'd34)) >> 9);
    18 : data_out <= ((data_in * $signed(8'd21)) >> 9);
    19 : data_out <= ((data_in * $signed(8'd8)) >> 9);
    default : data_out <= 36'b0;
  endcase
  end

else begin // IDLE state
  count <= 0;
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end


end

endmodule
```

```
module filter4(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
```

```
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (count)
    14 : data_out <= ((data_in * $signed(8'd3)) >> 9);
    15 : data_out <= ((data_in * $signed(8'd15)) >> 9);
    16 : data_out <= ((data_in * $signed(8'd27)) >> 9);
    17 : data_out <= ((data_in * $signed(8'd38)) >> 9);
    18 : data_out <= ((data_in * $signed(8'd50)) >> 9);
    19 : data_out <= ((data_in * $signed(8'd62)) >> 9);
    20 : data_out <= ((data_in * $signed(8'd64)) >> 9);
    21 : data_out <= ((data_in * $signed(8'd54)) >> 9);
    22 : data_out <= ((data_in * $signed(8'd43)) >> 9);
    23 : data_out <= ((data_in * $signed(8'd33)) >> 9);
    24 : data_out <= ((data_in * $signed(8'd23)) >> 9);
    25 : data_out <= ((data_in * $signed(8'd12)) >> 9);
    26 : data_out <= ((data_in * $signed(8'd2)) >> 9);
    default : data_out <= 36'b0;
  endcase
  end

else begin // IDLE state
  count <= 0;
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end


end

endmodule
```

```
module filter5(clock, reset, start, index_in, data_in, index_out, data_out, done);
```

```
input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    20 : data_out <= ((data_in * $signed(8'd4)) >> 9);
    21 : data_out <= ((data_in * $signed(8'd13)) >> 9);
    22 : data_out <= ((data_in * $signed(8'd22)) >> 9);
    23 : data_out <= ((data_in * $signed(8'd31)) >> 9);
```

```
    24 : data_out <= ((data_in * $signed(8'd40)) >> 9);
    25 : data_out <= ((data_in * $signed(8'd49)) >> 9);
    26 : data_out <= ((data_in * $signed(8'd58)) >> 9);
    27 : data_out <= ((data_in * $signed(8'd54)) >> 9);
    28 : data_out <= ((data_in * $signed(8'd46)) >> 9);
    29 : data_out <= ((data_in * $signed(8'd38)) >> 9);
    30 : data_out <= ((data_in * $signed(8'd30)) >> 9);
    31 : data_out <= ((data_in * $signed(8'd22)) >> 9);
    32 : data_out <= ((data_in * $signed(8'd14)) >> 9);
    33 : data_out <= ((data_in * $signed(8'd6)) >> 9);
    default : data_out <= 36'b0;
  endcase

  end

else begin // IDLE state
  count <= 0;
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end



end

endmodule
```

```
module filter6(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient
```

```
output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    27 : data_out <= ((data_in * $signed(8'd6)) >> 9);
    28 : data_out <= ((data_in * $signed(8'd13)) >> 9);
    29 : data_out <= ((data_in * $signed(8'd20)) >> 9);
    30 : data_out <= ((data_in * $signed(8'd27)) >> 9);
    31 : data_out <= ((data_in * $signed(8'd34)) >> 9);
    32 : data_out <= ((data_in * $signed(8'd41)) >> 9);
    33 : data_out <= ((data_in * $signed(8'd48)) >> 9);
    34 : data_out <= ((data_in * $signed(8'd51)) >> 9);
```

```
      35 : data_out <= ((data_in * $signed(8'd45)) >> 9);
      36 : data_out <= ((data_in * $signed(8'd39)) >> 9);
      37 : data_out <= ((data_in * $signed(8'd33)) >> 9);
      38 : data_out <= ((data_in * $signed(8'd27)) >> 9);
      39 : data_out <= ((data_in * $signed(8'd20)) >> 9);
      40 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      41 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      42 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


end

endmodule
```

```
module filter7(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
```

```
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    34 : data_out <= ((data_in * $signed(8'd1)) >> 9);
    35 : data_out <= ((data_in * $signed(8'd7)) >> 9);
    36 : data_out <= ((data_in * $signed(8'd12)) >> 9);
    37 : data_out <= ((data_in * $signed(8'd18)) >> 9);
    38 : data_out <= ((data_in * $signed(8'd23)) >> 9);
    39 : data_out <= ((data_in * $signed(8'd29)) >> 9);
    40 : data_out <= ((data_in * $signed(8'd34)) >> 9);
    41 : data_out <= ((data_in * $signed(8'd40)) >> 9);
    42 : data_out <= ((data_in * $signed(8'd45)) >> 9);
    43 : data_out <= ((data_in * $signed(8'd43)) >> 9);
```

```
    44 : data_out <= ((data_in * $signed(8'd39)) >> 9);
    45 : data_out <= ((data_in * $signed(8'd34)) >> 9);
    46 : data_out <= ((data_in * $signed(8'd29)) >> 9);
    47 : data_out <= ((data_in * $signed(8'd24)) >> 9);
    48 : data_out <= ((data_in * $signed(8'd19)) >> 9);
    49 : data_out <= ((data_in * $signed(8'd14)) >> 9);
    50 : data_out <= ((data_in * $signed(8'd10)) >> 9);
    51 : data_out <= ((data_in * $signed(8'd5)) >> 9);
    default : data_out <= 36'b0;
  endcase
  end

else begin // IDLE state
  count <= 0;
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end


end

endmodule
```

```
module filter8(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
```

```
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   data_out <= 36'b0;
   state <= IDLE;
   done <= NO;
   count <= 0;
   end

else if (start) begin
   state <= PROCESSING;
   done <= YES;
   count <= 0;
   end

else if (state == PROCESSING) begin
   done <= NO;
   count <= count + 1;
   if (count == 2047) state <= IDLE;
   else state <= state;
   index_out <= index_in;
   case (index_in)
      43 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      44 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      45 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      46 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      47 : data_out <= ((data_in * $signed(8'd20)) >> 9);
      48 : data_out <= ((data_in * $signed(8'd24)) >> 9);
      48 : data_out <= ((data_in * $signed(8'd29)) >> 9);
      49 : data_out <= ((data_in * $signed(8'd33)) >> 9);
      50 : data_out <= ((data_in * $signed(8'd37)) >> 9);
      51 : data_out <= ((data_in * $signed(8'd41)) >> 9);
      52 : data_out <= ((data_in * $signed(8'd37)) >> 9);
      53 : data_out <= ((data_in * $signed(8'd34)) >> 9);
      54 : data_out <= ((data_in * $signed(8'd30)) >> 9);
      55 : data_out <= ((data_in * $signed(8'd26)) >> 9);
```

```
    56 : data_out <= ((data_in * $signed(8'd22)) >> 9);
    57 : data_out <= ((data_in * $signed(8'd19)) >> 9);
    58 : data_out <= ((data_in * $signed(8'd15)) >> 9);
    59 : data_out <= ((data_in * $signed(8'd11)) >> 9);
    60 : data_out <= ((data_in * $signed(8'd7)) >> 9);
    61 : data_out <= ((data_in * $signed(8'd4)) >> 9);
    default : data_out <= 36'b0;
  endcase
  end

else begin // IDLE state
  count <= 0;
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end


end

endmodule
```

```
module filter9(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
```

```
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    52 : data_out <= ((data_in * $signed(8'd0)) >> 9);
    53 : data_out <= ((data_in * $signed(8'd3)) >> 9);
    54 : data_out <= ((data_in * $signed(8'd7)) >> 9);
    55 : data_out <= ((data_in * $signed(8'd10)) >> 9);
    56 : data_out <= ((data_in * $signed(8'd13)) >> 9);
    57 : data_out <= ((data_in * $signed(8'd17)) >> 9);
    58 : data_out <= ((data_in * $signed(8'd20)) >> 9);
    58 : data_out <= ((data_in * $signed(8'd23)) >> 9);
    59 : data_out <= ((data_in * $signed(8'd27)) >> 9);
    60 : data_out <= ((data_in * $signed(8'd30)) >> 9);
    61 : data_out <= ((data_in * $signed(8'd33)) >> 9);
    62 : data_out <= ((data_in * $signed(8'd36)) >> 9);
    63 : data_out <= ((data_in * $signed(8'd33)) >> 9);
    64 : data_out <= ((data_in * $signed(8'd30)) >> 9);
    65 : data_out <= ((data_in * $signed(8'd28)) >> 9);
```

```
    66 : data_out <= ((data_in * $signed(8'd25)) >> 9);
    67 : data_out <= ((data_in * $signed(8'd22)) >> 9);
    68 : data_out <= ((data_in * $signed(8'd19)) >> 9);
    69 : data_out <= ((data_in * $signed(8'd16)) >> 9);
    70 : data_out <= ((data_in * $signed(8'd13)) >> 9);
    71 : data_out <= ((data_in * $signed(8'd10)) >> 9);
    72 : data_out <= ((data_in * $signed(8'd7)) >> 9);
    73 : data_out <= ((data_in * $signed(8'd4)) >> 9);
    74 : data_out <= ((data_in * $signed(8'd1)) >> 9);
    default : data_out <= 36'b0;
  endcase
  end

else begin // IDLE state
  count <= 0;
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end


end

endmodule
```

```
module filter10(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient
```

```
output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    63 : data_out <= ((data_in * $signed(8'd0)) >> 9);
    64 : data_out <= ((data_in * $signed(8'd3)) >> 9);
    65 : data_out <= ((data_in * $signed(8'd5)) >> 9);
    66 : data_out <= ((data_in * $signed(8'd8)) >> 9);
    67 : data_out <= ((data_in * $signed(8'd10)) >> 9);
    68 : data_out <= ((data_in * $signed(8'd13)) >> 9);
    69 : data_out <= ((data_in * $signed(8'd16)) >> 9);
    70 : data_out <= ((data_in * $signed(8'd18)) >> 9);
    71 : data_out <= ((data_in * $signed(8'd21)) >> 9);
```

```
      72 : data_out <= ((data_in * $signed(8'd23)) >> 9);
      73 : data_out <= ((data_in * $signed(8'd26)) >> 9);
      74 : data_out <= ((data_in * $signed(8'd28)) >> 9);
      75 : data_out <= ((data_in * $signed(8'd31)) >> 9);
      76 : data_out <= ((data_in * $signed(8'd31)) >> 9);
      77 : data_out <= ((data_in * $signed(8'd29)) >> 9);
      78 : data_out <= ((data_in * $signed(8'd26)) >> 9);
      79 : data_out <= ((data_in * $signed(8'd24)) >> 9);
      80 : data_out <= ((data_in * $signed(8'd22)) >> 9);
      81 : data_out <= ((data_in * $signed(8'd19)) >> 9);
      82 : data_out <= ((data_in * $signed(8'd17)) >> 9);
      83 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      84 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      85 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      86 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      87 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      88 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      89 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

  else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


  end

  endmodule
```

```
module filter11(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
```

```
      76 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      77 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      78 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      79 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      80 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      81 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      82 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      83 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      84 : data_out <= ((data_in * $signed(8'd17)) >> 9);
      85 : data_out <= ((data_in * $signed(8'd19)) >> 9);
      86 : data_out <= ((data_in * $signed(8'd21)) >> 9);
      87 : data_out <= ((data_in * $signed(8'd23)) >> 9);
      88 : data_out <= ((data_in * $signed(8'd25)) >> 9);
      89 : data_out <= ((data_in * $signed(8'd27)) >> 9);
      90 : data_out <= ((data_in * $signed(8'd28)) >> 9);
      91 : data_out <= ((data_in * $signed(8'd26)) >> 9);
      92 : data_out <= ((data_in * $signed(8'd24)) >> 9);
      93 : data_out <= ((data_in * $signed(8'd22)) >> 9);
      94 : data_out <= ((data_in * $signed(8'd20)) >> 9);
      95 : data_out <= ((data_in * $signed(8'd19)) >> 9);
      96 : data_out <= ((data_in * $signed(8'd17)) >> 9);
      97 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      98 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      99 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      100 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      101 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      102 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      103 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      104 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      105 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

  else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


  end

  endmodule
```

```
module filter12(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   data_out <= 36'b0;
   state <= IDLE;
   done <= NO;
   count <= 0;
   end

else if (start) begin
```

```
    state <= PROCESSING;
    done <= YES;
    count <= 0;
    end

  else if (state == PROCESSING) begin
    done <= NO;
    count <= count + 1;
    if (count == 2047) state <= IDLE;
    else state <= state;
    index_out <= index_in;
    case (index_in)
      90 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      91 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      92 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      93 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      94 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      95 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      96 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      97 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      98 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      99 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      100 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      101 : data_out <= ((data_in * $signed(8'd18)) >> 9);
      102 : data_out <= ((data_in * $signed(8'd19)) >> 9);
      103 : data_out <= ((data_in * $signed(8'd21)) >> 9);
      104 : data_out <= ((data_in * $signed(8'd22)) >> 9);
      105 : data_out <= ((data_in * $signed(8'd24)) >> 9);
      106 : data_out <= ((data_in * $signed(8'd24)) >> 9);
      107 : data_out <= ((data_in * $signed(8'd23)) >> 9);
      108 : data_out <= ((data_in * $signed(8'd22)) >> 9);
      109 : data_out <= ((data_in * $signed(8'd20)) >> 9);
      110 : data_out <= ((data_in * $signed(8'd19)) >> 9);
      111 : data_out <= ((data_in * $signed(8'd18)) >> 9);
      112 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      113 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      114 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      115 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      116 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      117 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      118 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      119 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      120 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      121 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      122 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      123 : data_out <= ((data_in * $signed(8'd1)) >> 9);
```

```
      default : data_out <= 36'b0;
   endcase
   end

else begin // IDLE state
   count <= 0;
   index_out <= index_out;
   data_out <= data_out;
   done <= NO;
   end


end

endmodule
```

```
module filter13(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;
```

```
always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   data_out <= 36'b0;
   state <= IDLE;
   done <= NO;
   count <= 0;
   end

else if (start) begin
   state <= PROCESSING;
   done <= YES;
   count <= 0;
   end

else if (state == PROCESSING) begin
   done <= NO;
   count <= count + 1;
   if (count == 2047) state <= IDLE;
   else state <= state;
   index_out <= index_in;
   case (index_in)
      106 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      107 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      108 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      109 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      110 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      111 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      112 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      113 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      114 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      115 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      116 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      117 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      118 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      119 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      120 : data_out <= ((data_in * $signed(8'd17)) >> 9);
      121 : data_out <= ((data_in * $signed(8'd19)) >> 9);
      122 : data_out <= ((data_in * $signed(8'd20)) >> 9);
      123 : data_out <= ((data_in * $signed(8'd21)) >> 9);
      124 : data_out <= ((data_in * $signed(8'd22)) >> 9);
      125 : data_out <= ((data_in * $signed(8'd21)) >> 9);
      126 : data_out <= ((data_in * $signed(8'd20)) >> 9);
```

```
      127 : data_out <= ((data_in * $signed(8'd19)) >> 9);
      128 : data_out <= ((data_in * $signed(8'd17)) >> 9);
      129 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      130 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      131 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      132 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      133 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      134 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      135 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      136 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      137 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      138 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      139 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      140 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      141 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      142 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      143 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      144 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

  else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


  end

  endmodule
```

```
module filter14(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    124 : data_out <= ((data_in * $signed(8'd0)) >> 9);
```

```
  125 : data_out <= ((data_in * $signed(8'd1)) >> 9);
  126 : data_out <= ((data_in * $signed(8'd2)) >> 9);
  127 : data_out <= ((data_in * $signed(8'd3)) >> 9);
  128 : data_out <= ((data_in * $signed(8'd4)) >> 9);
  129 : data_out <= ((data_in * $signed(8'd5)) >> 9);
  130 : data_out <= ((data_in * $signed(8'd6)) >> 9);
  131 : data_out <= ((data_in * $signed(8'd7)) >> 9);
  132 : data_out <= ((data_in * $signed(8'd8)) >> 9);
  133 : data_out <= ((data_in * $signed(8'd9)) >> 9);
  134 : data_out <= ((data_in * $signed(8'd10)) >> 9);
  135 : data_out <= ((data_in * $signed(8'd11)) >> 9);
  136 : data_out <= ((data_in * $signed(8'd11)) >> 9);
  137 : data_out <= ((data_in * $signed(8'd12)) >> 9);
  138 : data_out <= ((data_in * $signed(8'd13)) >> 9);
  139 : data_out <= ((data_in * $signed(8'd14)) >> 9);
  140 : data_out <= ((data_in * $signed(8'd15)) >> 9);
  141 : data_out <= ((data_in * $signed(8'd16)) >> 9);
  142 : data_out <= ((data_in * $signed(8'd17)) >> 9);
  143 : data_out <= ((data_in * $signed(8'd18)) >> 9);
  144 : data_out <= ((data_in * $signed(8'd19)) >> 9);
  145 : data_out <= ((data_in * $signed(8'd19)) >> 9);
  146 : data_out <= ((data_in * $signed(8'd18)) >> 9);
  147 : data_out <= ((data_in * $signed(8'd17)) >> 9);
  148 : data_out <= ((data_in * $signed(8'd16)) >> 9);
  149 : data_out <= ((data_in * $signed(8'd16)) >> 9);
  150 : data_out <= ((data_in * $signed(8'd15)) >> 9);
  151 : data_out <= ((data_in * $signed(8'd14)) >> 9);
  152 : data_out <= ((data_in * $signed(8'd13)) >> 9);
  153 : data_out <= ((data_in * $signed(8'd12)) >> 9);
  154 : data_out <= ((data_in * $signed(8'd11)) >> 9);
  155 : data_out <= ((data_in * $signed(8'd11)) >> 9);
  156 : data_out <= ((data_in * $signed(8'd10)) >> 9);
  157 : data_out <= ((data_in * $signed(8'd9)) >> 9);
  158 : data_out <= ((data_in * $signed(8'd8)) >> 9);
  159 : data_out <= ((data_in * $signed(8'd7)) >> 9);
  160 : data_out <= ((data_in * $signed(8'd6)) >> 9);
  161 : data_out <= ((data_in * $signed(8'd6)) >> 9);
  162 : data_out <= ((data_in * $signed(8'd5)) >> 9);
  163 : data_out <= ((data_in * $signed(8'd4)) >> 9);
  164 : data_out <= ((data_in * $signed(8'd3)) >> 9);
  165 : data_out <= ((data_in * $signed(8'd2)) >> 9);
  166 : data_out <= ((data_in * $signed(8'd2)) >> 9);
  167 : data_out <= ((data_in * $signed(8'd1)) >> 9);
  default : data_out <= 36'b0;
endcase
end
```

```
else begin // IDLE state
   count <= 0;
   index_out <= index_out;
   data_out <= data_out;
   done <= NO;
   end


end

endmodule
```

```
module filter15(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
```

```
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
  data_out <= 36'b0;
  state <= IDLE;
  done <= NO;
  count <= 0;
  end

else if (start) begin
  state <= PROCESSING;
  done <= YES;
  count <= 0;
  end

else if (state == PROCESSING) begin
  done <= NO;
  count <= count + 1;
  if (count == 2047) state <= IDLE;
  else state <= state;
  index_out <= index_in;
  case (index_in)
    145 : data_out <= ((data_in * $signed(8'd0)) >> 9);
    146 : data_out <= ((data_in * $signed(8'd1)) >> 9);
    147 : data_out <= ((data_in * $signed(8'd2)) >> 9);
    148 : data_out <= ((data_in * $signed(8'd3)) >> 9);
    149 : data_out <= ((data_in * $signed(8'd3)) >> 9);
    150 : data_out <= ((data_in * $signed(8'd4)) >> 9);
    151 : data_out <= ((data_in * $signed(8'd5)) >> 9);
    152 : data_out <= ((data_in * $signed(8'd6)) >> 9);
    153 : data_out <= ((data_in * $signed(8'd6)) >> 9);
    154 : data_out <= ((data_in * $signed(8'd7)) >> 9);
    155 : data_out <= ((data_in * $signed(8'd8)) >> 9);
    156 : data_out <= ((data_in * $signed(8'd8)) >> 9);
    157 : data_out <= ((data_in * $signed(8'd9)) >> 9);
    158 : data_out <= ((data_in * $signed(8'd10)) >> 9);
    159 : data_out <= ((data_in * $signed(8'd11)) >> 9);
    160 : data_out <= ((data_in * $signed(8'd11)) >> 9);
    161 : data_out <= ((data_in * $signed(8'd12)) >> 9);
    162 : data_out <= ((data_in * $signed(8'd13)) >> 9);
```

```
      163 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      164 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      165 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      166 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      167 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      168 : data_out <= ((data_in * $signed(8'd17)) >> 9);
      169 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      170 : data_out <= ((data_in * $signed(8'd16)) >> 9);
      171 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      172 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      173 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      174 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      175 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      176 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      177 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      178 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      179 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      180 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      181 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      182 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      183 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      184 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      185 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      186 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      187 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      188 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      189 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      190 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      191 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      192 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      193 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      194 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

  else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


  end
```

```
endmodule
```

```
module filter16(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   data_out <= 36'b0;
   state <= IDLE;
   done <= NO;
```

```
    count <= 0;
    end

  else if (start) begin
    state <= PROCESSING;
    done <= YES;
    count <= 0;
    end

  else if (state == PROCESSING) begin
    done <= NO;
    count <= count + 1;
    if (count == 2047) state <= IDLE;
    else state <= state;
    index_out <= index_in;
    case (index_in)
      168 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      169 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      170 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      171 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      172 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      173 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      174 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      175 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      176 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      177 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      178 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      179 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      180 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      181 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      182 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      183 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      184 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      185 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      186 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      187 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      188 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      189 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      190 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      191 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      192 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      193 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      194 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      195 : data_out <= ((data_in * $signed(8'd15)) >> 9);
      196 : data_out <= ((data_in * $signed(8'd14)) >> 9);
      197 : data_out <= ((data_in * $signed(8'd14)) >> 9);
```

```
      198 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      199 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      200 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      201 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      202 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      203 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      204 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      205 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      206 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      207 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      208 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      209 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      210 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      211 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      212 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      213 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      214 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      215 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      216 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      217 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      218 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      219 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      220 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      221 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      222 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      223 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      224 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

  else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


  end

  endmodule
```

```
module filter17(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   data_out <= 36'b0;
   state <= IDLE;
   done <= NO;
   count <= 0;
```

```
    end

  else if (start) begin
    state <= PROCESSING;
    done <= YES;
    count <= 0;
    end

  else if (state == PROCESSING) begin
    done <= NO;
    count <= count + 1;
    if (count == 2047) state <= IDLE;
    else state <= state;
    index_out <= index_in;
    case (index_in)
      195 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      196 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      197 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      198 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      199 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      200 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      201 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      202 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      203 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      204 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      205 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      206 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      207 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      208 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      209 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      210 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      211 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      212 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      213 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      214 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      215 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      216 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      217 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      218 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      219 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      220 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      221 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      222 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      223 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      224 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      225 : data_out <= ((data_in * $signed(8'd13)) >> 9);
```

```
      226 : data_out <= ((data_in * $signed(8'd13)) >> 9);
      227 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      228 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      229 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      230 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      231 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      232 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      233 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      234 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      235 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      236 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      237 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      238 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      239 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      240 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      241 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      242 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      243 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      244 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      245 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      246 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      247 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      248 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      249 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      250 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      251 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      252 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      253 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      254 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      255 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      256 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      257 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      258 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

  else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


  end
```

```
endmodule
```

```
module filter18(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
   index_out <= 0;
   data_out <= 36'b0;
   state <= IDLE;
   done <= NO;
   count <= 0;
   end
```

```
else if (start) begin
   state <= PROCESSING;
   done <= YES;
   count <= 0;
   end

else if (state == PROCESSING) begin
   done <= NO;
   count <= count + 1;
   if (count == 2047) state <= IDLE;
   else state <= state;
   index_out <= index_in;
   case (index_in)
      225 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      226 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      227 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      228 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      229 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      230 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      231 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      232 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      233 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      234 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      235 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      236 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      237 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      238 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      239 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      240 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      241 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      242 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      243 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      244 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      245 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      246 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      247 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      248 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      249 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      250 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      251 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      252 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      253 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      254 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      255 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      256 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      257 : data_out <= ((data_in * $signed(8'd11)) >> 9);
```

```
      258 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      259 : data_out <= ((data_in * $signed(8'd12)) >> 9);
      260 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      261 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      262 : data_out <= ((data_in * $signed(8'd11)) >> 9);
      263 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      264 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      265 : data_out <= ((data_in * $signed(8'd10)) >> 9);
      266 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      267 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      268 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      269 : data_out <= ((data_in * $signed(8'd9)) >> 9);
      270 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      271 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      272 : data_out <= ((data_in * $signed(8'd8)) >> 9);
      273 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      274 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      275 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      276 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      277 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      278 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      279 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      280 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      281 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      282 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      283 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      284 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      285 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      286 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      287 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      288 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      289 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      290 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      291 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      292 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      293 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      294 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      295 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      296 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      297 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

  else begin // IDLE state
    count <= 0;
```

```
  index_out <= index_out;
  data_out <= data_out;
  done <= NO;
  end


end

endmodule
```

```
module filter19(clock, reset, start, index_in, data_in, index_out, data_out, done);

input [10:0] index_in;
input clock, reset, start;
input signed [36:0] data_in; // assuming power is 39-bit wide for each coefficient

output done;
output [10:0] index_out;
output signed [35:0] data_out;

reg [10:0] index_out, count;
reg signed [35:0] data_out;
reg done, state;

parameter NO = 0;
parameter YES = 1;
parameter PROCESSING = 1;
parameter IDLE = 0;


always @(posedge clock) begin

if (reset) begin
  index_out <= 0;
```

```
    data_out <= 36'b0;
    state <= IDLE;
    done <= NO;
    count <= 0;
    end

  else if (start) begin
    state <= PROCESSING;
    done <= YES;
    count <= 0;
    end

  else if (state == PROCESSING) begin
    done <= NO;
    count <= count + 1;
    if (count == 2047) state <= IDLE;
    else state <= state;
    index_out <= index_in;
    case (index_in)
      259 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      260 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      261 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      262 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      263 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      264 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      265 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      266 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      267 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      268 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      269 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      270 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      271 : data_out <= ((data_in * $signed(8'd3)) >> 9);
      272 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      273 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      274 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      275 : data_out <= ((data_in * $signed(8'd4)) >> 9);
      276 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      277 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      278 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      279 : data_out <= ((data_in * $signed(8'd5)) >> 9);
      280 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      281 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      282 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      283 : data_out <= ((data_in * $signed(8'd6)) >> 9);
      284 : data_out <= ((data_in * $signed(8'd7)) >> 9);
      285 : data_out <= ((data_in * $signed(8'd7)) >> 9);
```

```
286 : data_out <= ((data_in * $signed(8'd7)) >> 9);
287 : data_out <= ((data_in * $signed(8'd8)) >> 9);
288 : data_out <= ((data_in * $signed(8'd8)) >> 9);
289 : data_out <= ((data_in * $signed(8'd8)) >> 9);
290 : data_out <= ((data_in * $signed(8'd8)) >> 9);
291 : data_out <= ((data_in * $signed(8'd9)) >> 9);
292 : data_out <= ((data_in * $signed(8'd9)) >> 9);
293 : data_out <= ((data_in * $signed(8'd9)) >> 9);
294 : data_out <= ((data_in * $signed(8'd9)) >> 9);
295 : data_out <= ((data_in * $signed(8'd10)) >> 9);
296 : data_out <= ((data_in * $signed(8'd10)) >> 9);
297 : data_out <= ((data_in * $signed(8'd10)) >> 9);
298 : data_out <= ((data_in * $signed(8'd10)) >> 9);
299 : data_out <= ((data_in * $signed(8'd10)) >> 9);
300 : data_out <= ((data_in * $signed(8'd10)) >> 9);
301 : data_out <= ((data_in * $signed(8'd9)) >> 9);
302 : data_out <= ((data_in * $signed(8'd9)) >> 9);
303 : data_out <= ((data_in * $signed(8'd9)) >> 9);
304 : data_out <= ((data_in * $signed(8'd9)) >> 9);
305 : data_out <= ((data_in * $signed(8'd9)) >> 9);
306 : data_out <= ((data_in * $signed(8'd8)) >> 9);
307 : data_out <= ((data_in * $signed(8'd8)) >> 9);
308 : data_out <= ((data_in * $signed(8'd8)) >> 9);
309 : data_out <= ((data_in * $signed(8'd8)) >> 9);
310 : data_out <= ((data_in * $signed(8'd7)) >> 9);
311 : data_out <= ((data_in * $signed(8'd7)) >> 9);
312 : data_out <= ((data_in * $signed(8'd7)) >> 9);
313 : data_out <= ((data_in * $signed(8'd7)) >> 9);
314 : data_out <= ((data_in * $signed(8'd6)) >> 9);
315 : data_out <= ((data_in * $signed(8'd6)) >> 9);
316 : data_out <= ((data_in * $signed(8'd6)) >> 9);
317 : data_out <= ((data_in * $signed(8'd6)) >> 9);
318 : data_out <= ((data_in * $signed(8'd5)) >> 9);
319 : data_out <= ((data_in * $signed(8'd5)) >> 9);
320 : data_out <= ((data_in * $signed(8'd5)) >> 9);
321 : data_out <= ((data_in * $signed(8'd5)) >> 9);
322 : data_out <= ((data_in * $signed(8'd5)) >> 9);
323 : data_out <= ((data_in * $signed(8'd4)) >> 9);
324 : data_out <= ((data_in * $signed(8'd4)) >> 9);
325 : data_out <= ((data_in * $signed(8'd4)) >> 9);
326 : data_out <= ((data_in * $signed(8'd4)) >> 9);
327 : data_out <= ((data_in * $signed(8'd3)) >> 9);
328 : data_out <= ((data_in * $signed(8'd3)) >> 9);
329 : data_out <= ((data_in * $signed(8'd3)) >> 9);
330 : data_out <= ((data_in * $signed(8'd3)) >> 9);
331 : data_out <= ((data_in * $signed(8'd2)) >> 9);
```

```
      332 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      333 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      334 : data_out <= ((data_in * $signed(8'd2)) >> 9);
      335 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      336 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      337 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      338 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      339 : data_out <= ((data_in * $signed(8'd1)) >> 9);
      340 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      341 : data_out <= ((data_in * $signed(8'd0)) >> 9);
      default : data_out <= 36'b0;
    endcase
    end

else begin // IDLE state
    count <= 0;
    index_out <= index_out;
    data_out <= data_out;
    done <= NO;
    end


end

endmodule
```

```
module accumulator(clock, reset, start, index_in, data_in, data_out, time_int, done);

input clock, reset, start;
input [10:0] index_in;
input signed [35:0] data_in;

output done;
output [9:0] time_int;
output signed [62:0] data_out;
```

```
parameter NO = 0;
parameter YES = 1;
parameter ACCUMULATING = 1;
parameter IDLE = 0;

reg done;
reg [9:0] time_int, time_count;
reg [62:0] data_out;
reg signed [62:0] temp_data;
reg state;

always @(posedge clock) begin

if (reset) begin
            data_out <= 0;
            temp_data <= 0;
            done <= NO;
            time_int <= 0;
            time_count <= 0;
            state <= IDLE;
            end
else if (start) begin
            state <= ACCUMULATING;
            temp_data <= 0;
            done <= NO;
            end
else if (state == ACCUMULATING) begin
  done <= NO;
            if (index_in == 2047) begin
              state <= IDLE;
                        time_int <= time_count;
                        time_count <= time_count + 1;
                        done <= YES;
                        end
            else begin
              state <= state;
                        time_int <= time_int;
                        time_count <= time_count;
                        end
            temp_data <= temp_data + data_in;
            end

else begin // IDLE state
  done <= NO;
            time_int <= 0;
```

```
            time_count <= 0;
            state <= state;
            data_out <= temp_data[62:0];
            end

end

endmodule
```

```
module lg(clock,reset,compute,data_in,data_out,computed);

input clock, reset, compute;
input signed [62:0] data_in; // 2's complement from previous module

output [16:0] data_out; // output is in binary magnitude format
output computed;

parameter YES = 1;
parameter IDLE = 2'b00;
parameter COMP_INDEX = 2'b01;
parameter COMP_OUT = 2'b10;
parameter NO = 0;


reg [16:0] data_out, index;
reg [3:0] decimals;
reg [1:0] state;
reg computed;


wire [62:0] temp_data;
assign temp_data = {1'b0, data_in[61:0]}; // convert data_in to binary magnitude format

// note that this last conversion follows from the fact that the input must be
```

```
// positive because it was gotten from (abs(fft))^2, and then passed through the
// Mel filters (multiplied by positive numbers)


always @(posedge clock) begin

if (reset) begin
  index <= 0;
  data_out <= 17'b0;
  state <= IDLE;
          computed <= NO;
  end

else if (compute == YES) begin
  state <= COMP_INDEX;
  index <= 0;
  data_out <= data_out;
          computed <= NO;
  end

else begin

  case (state)

  IDLE : begin
    state <= IDLE;
    index <= 0;
    data_out <= data_out;
                    computed <= NO;
    end

  COMP_INDEX: begin

                    if (temp_data[62] == 1) begin
      index <= 62000;
      decimals <= temp_data[61:58];
      end
    else if (temp_data[61] == 1) begin
      index <= 61000;
      decimals <= temp_data[60:57];
      end
    else if (temp_data[60] == 1) begin
      index <= 60000;
      decimals <= temp_data[59:56];
      end
    else if (temp_data[59] == 1) begin
```

```
    index <= 59000;
    decimals <= temp_data[58:55];
    end
  else if (temp_data[58] == 1) begin
    index <= 58000;
    decimals <= temp_data[57:54];
    end
  else if (temp_data[57] == 1) begin
    index <= 57000;
    decimals <= temp_data[56:53];
    end
  else if (temp_data[56] == 1) begin
    index <= 56000;
    decimals <= temp_data[55:52];
    end
  else if (temp_data[55] == 1) begin
    index <= 55000;
    decimals <= temp_data[54:51];
    end
  else if (temp_data[54] == 1) begin
    index <= 54000;
    decimals <= temp_data[53:50];
    end
  else if (temp_data[53] == 1) begin
    index <= 53000;
    decimals <= temp_data[52:49];
    end
  else if (temp_data[52] == 1) begin
    index <= 52000;
    decimals <= temp_data[51:48];
    end
  else if (temp_data[51] == 1) begin
    index <= 51000;
    decimals <= temp_data[50:47];
    end
  else if (temp_data[50] == 1) begin
    index <= 50000;
    decimals <= temp_data[49:46];
    end
  else if (temp_data[49] == 1) begin
    index <= 49000;
    decimals <= temp_data[48:45];
    end
  else if (temp_data[48] == 1) begin
    index <= 48000;
    decimals <= temp_data[47:44];
```

```
   end
else if (temp_data[47] == 1) begin
   index <= 47000;
   decimals <= temp_data[46:43];
   end
else if (temp_data[46] == 1) begin
   index <= 46000;
   decimals <= temp_data[45:42];
   end
else if (temp_data[45] == 1) begin
   index <= 45000;
   decimals <= temp_data[44:41];
   end
else if (temp_data[44] == 1) begin
   index <= 44000;
   decimals <= temp_data[43:40];
   end
else if (temp_data[43] == 1) begin
   index <= 43000;
   decimals <= temp_data[42:39];
   end
else if (temp_data[42] == 1) begin
   index <= 42000;
   decimals <= temp_data[41:38];
   end
else if (temp_data[41] == 1) begin
   index <= 41000;
   decimals <= temp_data[40:37];
   end
else if (temp_data[40] == 1) begin
   index <= 40000;
   decimals <= temp_data[39:36];
   end
else if (temp_data[39] == 1) begin
   index <= 39000;
   decimals <= temp_data[38:35];
   end
else if (temp_data[38] == 1) begin
   index <= 38000;
   decimals <= temp_data[37:34];
   end
else if (temp_data[37] == 1) begin
   index <= 37000;
   decimals <= temp_data[36:33];
   end
else if (temp_data[36] == 1) begin
```

```
    index <= 36000;
    decimals <= temp_data[35:32];
    end
  else if (temp_data[35] == 1) begin
    index <= 35000;
    decimals <= temp_data[34:31];
    end
  else if (temp_data[34] == 1) begin
    index <= 34000;
    decimals <= temp_data[33:30];
    end
  else if (temp_data[33] == 1) begin
    index <= 33000;
    decimals <= temp_data[32:29];
    end
  else if (temp_data[32] == 1) begin
    index <= 32000;
    decimals <= temp_data[31:28];
    end
  else if (temp_data[31] == 1) begin
    index <= 31000;
    decimals <= temp_data[30:27];
    end
  else if (temp_data[30] == 1) begin
    index <= 30000;
    decimals <= temp_data[29:26];
    end
  else if (temp_data[29] == 1) begin
    index <= 29000;
    decimals <= temp_data[28:25];
    end
  else if (temp_data[28] == 1) begin
    index <= 28000;
    decimals <= temp_data[27:24];
    end
  else if (temp_data[27] == 1) begin
    index <= 27000;
    decimals <= temp_data[26:23];
    end
  else if (temp_data[26] == 1) begin
    index <= 26000;
    decimals <= temp_data[25:22];
    end
  else if (temp_data[25] == 1) begin
    index <= 25000;
    decimals <= temp_data[24:21];
```

```
    end
  else if (temp_data[24] == 1) begin
    index <= 24000;
    decimals <= temp_data[23:20];
    end
  else if (temp_data[23] == 1) begin
    index <= 23000;
    decimals <= temp_data[22:19];
    end
  else if (temp_data[22] == 1) begin
    index <= 22000;
    decimals <= temp_data[21:18];
    end
  else if (temp_data[21] == 1) begin
    index <= 21000;
    decimals <= temp_data[20:17];
    end
  else if (temp_data[20] == 1) begin
    index <= 20000;
    decimals <= temp_data[19:16];
    end
  else if (temp_data[19] == 1) begin
    index <= 19000;
    decimals <= temp_data[18:15];
    end
  else if (temp_data[18] == 1) begin
    index <= 18000;
    decimals <= temp_data[17:14];
    end
  else if (temp_data[17] == 1) begin
    index <= 17000;
    decimals <= temp_data[16:13];
    end
  else if (temp_data[16] == 1) begin
    index <= 16000;
    decimals <= temp_data[15:12];
    end
  else if (temp_data[15] == 1) begin
    index <= 15000;
    decimals <= temp_data[14:11];
    end
  else if (temp_data[14] == 1) begin
    index <= 14000;
    decimals <= temp_data[13:10];
    end
  else if (temp_data[13] == 1) begin
```

```
    index <= 13000;
    decimals <= temp_data[12:9];
    end
  else if (temp_data[12] == 1) begin
    index <= 12000;
    decimals <= temp_data[11:8];
    end
  else if (temp_data[11] == 1) begin
    index <= 11000;
    decimals <= temp_data[10:7];
    end
  else if (temp_data[10] == 1) begin
    index <= 10000;
    decimals <= temp_data[9:6];
    end
  else if (temp_data[9] == 1) begin
    index <= 9000;
    decimals <= temp_data[8:5];
    end
  else if (temp_data[8] == 1) begin
    index <= 8000;
    decimals <= temp_data[7:4];
    end
  else if (temp_data[7] == 1) begin
    index <= 7000;
    decimals <= temp_data[6:3];
    end
  else if (temp_data[6] == 1) begin
    index <= 6000;
    decimals <= temp_data[5:2];
    end
  else if (temp_data[5] == 1) begin
    index <= 5000;
    decimals <= temp_data[4:1];
    end
  else if (temp_data[4] == 1) begin
    index <= 4000;
    decimals <= temp_data[3:0];
    end
  else if (temp_data[3] == 1) begin
    index <= 3000;
    decimals <= {temp_data[2:0], 1'b0};
    end
  else if (temp_data[2] == 1) begin
    index <= 2000;
    decimals <= {temp_data[1:0], 2'b0};
```

```
    end
  else if (temp_data[1] == 1) begin
    index <= 1000;
    decimals <= {temp_data[0], 3'b0};
    end
  else begin
    index <= 0;
    decimals <= 4'b0;
    end

  state <= COMP_OUT;

  end

COMP_OUT: begin

  case (decimals)

    4'd0 : data_out <= index + 10'd0;

    4'd1 : data_out <= index + 10'd87;

    4'd2 : data_out <= index + 10'd170;

    4'd3 : data_out <= index + 10'd245;

    4'd4 : data_out <= index + 10'd322;

    4'd5 : data_out <= index + 10'd392;

    4'd6 : data_out <= index + 10'd459;

    4'd7 : data_out <= index + 10'd524;

    4'd8 : data_out <= index + 10'd585;

    4'd9 : data_out <= index + 10'd644;

    4'd10 : data_out <= index + 10'd700;

    4'd11 : data_out <= index + 10'd755;

    4'd12 : data_out <= index + 10'd807;

    4'd13 : data_out <= index + 10'd857;
```

```
        4'd14 : data_out <= index + 10'd907;

        4'd15 : data_out <= index + 10'd954;

      endcase

      state <= IDLE;
                        computed <= YES;

      end

    default : state <= IDLE;

    endcase

    end

end

endmodule
```

```
module dct1(clock,reset,filter1,filter2,filter3,filter4,filter5,filter6,filter7,filter8,
          filter9 ,filter10,filter11,filter12,filter13,filter14,filter15,filter16,
          filter17,filter18,filter19,start,coef,done);

input clock, reset, start;
input signed [16:0] filter1,filter2,filter3,filter4,filter5,filter6,filter7,filter8,
                filter9 ,filter10,filter11,filter12,filter13,filter14,filter15,filter16,
                filter17,filter18,filter19;

output done;
output signed [45:0] coef;

parameter DCTP1 = 11'b00100010110;
parameter DCTP2 = 11'b01000101100;
```

```
parameter DCTP3 = 11'b01000101100;
parameter DCTP4 = 11'b01000101100;
parameter DCTP5 = 11'b01000101100;
parameter DCTP6 = 11'b01000101100;
parameter DCTP7 = 11'b01000101100;
parameter DCTP8 = 11'b01000101100;
parameter DCTP9 = 11'b01000101100;
parameter DCTP10 = 11'b01000101100;
parameter DCTP11 = 11'b01000101100;
parameter DCTP12 = 11'b01000101100;
parameter DCTP13 = 11'b01000101100;
parameter DCTP14 = 11'b01000101100;
parameter DCTP15 = 11'b01000101100;
parameter DCTP16 = 11'b01000101100;
parameter DCTP17 = 11'b01000101100;
parameter DCTP18 = 11'b01000101100;
parameter DCTP19 = 11'b00100010110;

parameter IDLE = 2'b00;
parameter WAIT_INPUT_READY = 2'b01;
parameter PUT_OUTPUT = 2'b10;
parameter SIGNAL_READINESS = 2'b11;
parameter OFF = 0;
parameter ON = 1;

reg done;
reg [1:0] state;
reg signed [45:0] coef;

wire signed [45:0] temp;
assign temp = ((filter1 * $signed(DCTP1)) + (filter2 * $signed(DCTP2)) + (filter3 *
$signed(DCTP3)) + (filter4 * $signed(DCTP4)) +
        (filter5 * $signed(DCTP5)) + (filter6 * $signed(DCTP6)) + (filter7 *
$signed(DCTP7)) + (filter8 * $signed(DCTP8)) +
        (filter9 * $signed(DCTP9)) + (filter10 * $signed(DCTP10)) + (filter11 *
$signed(DCTP11)) + (filter12 * $signed(DCTP12)) +
        (filter13 * $signed(DCTP13)) + (filter14 * $signed(DCTP14)) + (filter15 *
$signed(DCTP15)) + (filter16 * $signed(DCTP16)) +
        (filter17 * $signed(DCTP17)) + (filter18 * $signed(DCTP18)) + (filter19 *
$signed(DCTP19))) ;


always @(posedge clock) begin

if (reset) begin
  coef <= 0;
```

```
  state <= IDLE;
  done <= OFF;
  end
else if (start) begin
  coef <= coef;
  state <= WAIT_INPUT_READY;
  done <= OFF;
  end

else if (state == WAIT_INPUT_READY) begin
  coef <= coef;
  state <= PUT_OUTPUT;
  done <= OFF;
  end

else if (state == PUT_OUTPUT) begin
  coef <= temp[45:0];
  state <= SIGNAL_READINESS;
  done <= OFF;
  end

else if (state == SIGNAL_READINESS) begin
  coef <= coef;
  state <= IDLE;
  done <= ON;
  end

else begin // IDLE state
  coef <= coef;
  state <= state;
  done <= OFF;
  end

end

endmodule




module distance_processor(clock, reset, new_stored, new_current, endd, user,
                                  user1_out, user2_out, user3_out, user4_out, ready, cf_1,
cf_2, cf_3, cf_4,
```

```
                                cf_5, cf_6, cf_7, cf_8, cf_9, cf_10, cf_11, cf_12, cf_13,
cf_14, cf_15, cf_16,
                                pf_1, pf_2, pf_3, pf_4, pf_5, pf_6, pf_7, pf_8, pf_9, pf_10,
pf_11, pf_12,
                                pf_13, pf_14, pf_15, pf_16);

input signed [15:0] cf_1, cf_2, cf_3, cf_4, cf_5, cf_6, cf_7, cf_8, cf_9, cf_10, cf_11,
cf_12, cf_13, cf_14, cf_15, cf_16;
input signed [15:0] pf_1, pf_2, pf_3, pf_4, pf_5, pf_6, pf_7, pf_8, pf_9, pf_10, pf_11,
pf_12, pf_13, pf_14, pf_15, pf_16;
input clock, reset, new_stored, new_current, endd;
input [1:0] user;

output [63:0] user1_out, user2_out, user3_out, user4_out;
output ready;

parameter ON = 1;
parameter OFF = 0;
parameter COMPUTING = 0;
parameter ACCUMULATE = 1;
parameter CHECK_END = 2;
parameter IDLE = 3;

wire [22:0] distance;
wire done;
reg compute, give_output, ready;
reg [1:0] state, user_id;
reg [8:0] count, sample_count;
reg [63:0] user1_out, user2_out, user3_out, user4_out, user1, user2, user3, user4;
reg [22:0] user1_temp, user2_temp, user3_temp, user4_temp;

vector_dist dister(clock, reset, cf_1, cf_2, cf_3, cf_4, cf_5, cf_6, cf_7, cf_8, cf_9, cf_10,
cf_11,
                cf_12, cf_13, cf_14, cf_15, cf_16, compute, pf_1, pf_2, pf_3, pf_4, pf_5,
pf_6,
                pf_7, pf_8, pf_9, pf_10, pf_11, pf_12, pf_13, pf_14, pf_15, pf_16, done,
distance);



always @(posedge clock) begin

if (reset) begin
  user1_temp <= 0;
  user2_temp <= 0;
  user3_temp <= 0;
```

```
  user4_temp <= 0;
  user1 <= 0;
  user2 <= 0;
  user3 <= 0;
  user4 <= 0;
  count <= 0;
  state <= IDLE;
  compute <= OFF;
  give_output <= OFF;
  sample_count <= 0;
  ready <= OFF;
  end

else if (endd) give_output <= ON;

else if (new_current) begin
  sample_count <= sample_count + 1;
  count <= 0;
  user1_temp <= 0;
  user2_temp <= 0;
  user3_temp <= 0;
  user4_temp <= 0;
          end

else if (new_stored) begin
  user_id <= user;
  state <= COMPUTING;
  compute <= ON;
  end

else begin

  compute <= OFF;

  case (state)

    COMPUTING : state <= (done) ? ACCUMULATE : state;

    ACCUMULATE : begin

      count <= count + 1;
      case (user_id)
        0 : user1_temp <= (count == 0) ? distance :
                    (user1_temp <= distance) ? user1_temp : distance;
        1 : user2_temp <= (count == 0) ? distance :
                    (user2_temp <= distance) ? user2_temp : distance;
```

```
    2 : user3_temp <= (count == 0) ? distance :
                    (user3_temp <= distance) ? user3_temp : distance;
    3 : user4_temp <= (count == 0) ? distance :
                    (user4_temp <= distance) ? user4_temp : distance;
  endcase
  state <= CHECK_END;
  end

CHECK_END : begin

  if (count == 160) begin
    case (user_id)
      0 : user1 <= user1_temp + user1;
      1 : user2 <= user2_temp + user2;
      2 : user3 <= user3_temp + user3;
      3 : user4 <= user4_temp + user4;
    endcase
    count <= 0;
    end

  state <= IDLE;
  end



IDLE : begin

  if ((give_output) && (sample_count == 160)) begin
    user1_out <= user1;
    user2_out <= user2;
    user3_out <= user3;
    user4_out <= user4;
    user1 <= 0;
    user2 <= 0;
    user3 <= 0;
    user4 <= 0;
    sample_count <= 0;
    ready <= ON;
                                        give_output <= OFF;
    end
  else ready <= OFF;

  state <= state;

  end
```

```
  endcase
  end


end

endmodule
```

```
module vector_dist(clock, reset, cf_1, cf_2, cf_3, cf_4, cf_5, cf_6, cf_7, cf_8, cf_9,
cf_10, cf_11,
                   cf_12, cf_13, cf_14, cf_15, cf_16, compute, pf_1, pf_2, pf_3, pf_4, pf_5,
pf_6,
                   pf_7, pf_8, pf_9, pf_10, pf_11, pf_12, pf_13, pf_14, pf_15, pf_16,
computed, distance);

input clock, reset, compute;
input signed [15:0] cf_1, cf_2, cf_3, cf_4, cf_5, cf_6, cf_7, cf_8, cf_9, cf_10, cf_11,
cf_12, cf_13, cf_14, cf_15, cf_16;
input signed [15:0] pf_1, pf_2, pf_3, pf_4, pf_5, pf_6, pf_7, pf_8, pf_9, pf_10, pf_11,
pf_12, pf_13, pf_14, pf_15, pf_16;

// each cf_i or pf_i coefficient vector, 0 < i < 17, has 16 bits for each spectral coefficient.

output computed;
output [22:0] distance;

reg computing, computed;
reg signed [46:0] s_dist;
reg start_sqrt;
wire [45:0] dist;
wire [22:0] distance;
wire done;
```

```
parameter YES = 1'b1;
parameter NO = 1'b0;

assign dist = s_dist[45:0];    // s_dist is a distance measure and thus is positive, ie to get
unsigned distance we just
                        // drop the MSB (sign bit). dist has 46 bits in total.

sqrt dist_sqrt(clock, reset, start_sqrt, dist, distance, done);

always @(posedge clock) begin

if (reset == YES) begin
  computed <= NO;
  s_dist <= 0;
  computing <= NO;
            start_sqrt <= NO;
  end

else if (compute == YES) begin
  computed <= NO;
  s_dist <= ((cf_1 - pf_1) * (cf_1 - pf_1)) + ((cf_2 - pf_2) * (cf_2 - pf_2)) + ((cf_3 -
pf_3) * (cf_3 - pf_3)) +
            ((cf_4 - pf_4) * (cf_4 - pf_4)) + ((cf_5 - pf_5) * (cf_5 - pf_5)) + ((cf_6 - pf_6) *
(cf_6 - pf_6)) +
            ((cf_7 - pf_7) * (cf_7 - pf_7)) + ((cf_8 - pf_8) * (cf_8 - pf_8)) + ((cf_9 - pf_9) *
(cf_9 - pf_9)) +
            ((cf_10 - pf_10) * (cf_10 - pf_10)) + ((cf_11 - pf_11) * (cf_11 - pf_11)) +
((cf_12 - pf_12) * (cf_12 - pf_12)) +
            ((cf_13 - pf_13) * (cf_13 - pf_13)) + ((cf_14 - pf_14) * (cf_14 - pf_14)) +
((cf_15 - pf_15) * (cf_15 - pf_15)) +
            ((cf_16 - pf_16) * (cf_16 - pf_16));
  computing <= YES;
             start_sqrt <= YES;
  end

else if (computing == YES) begin
  start_sqrt <= NO;
  if (done) begin
                        computing <= NO;
                        computed <= YES;
                        end
            else begin
                        computing <= YES;
                        computed <= NO;
                        end
```

```
  s_dist <= s_dist;
  end

else begin
  computing <= computing;
  computed <= NO;
  s_dist <= s_dist;
            end

end

endmodule
```

```
module sqrt(clock, reset, start, x, sqrt_x, done);

input clock, reset, start;
input [45:0] x;
output done;
output [22:0] sqrt_x;

reg compute, done;
reg [22:0] sqrt_x;
reg [574:0] temp_num;
```

```
reg [528:0] temp_root;

reg [4:0] i;

always @(posedge clock) begin

if (reset) begin
  temp_num <= 0;
  temp_root <= 0;
  i <= 22;
  sqrt_x <= 0;
  compute <= 0;
  done <= 0;
  end

else if (start) begin
  compute <= 1;
  temp_num[574:550] <= {23'b0, x[45:44]};
  temp_root[528:506] <= 23'b0;
  i <= 22;
  end

else if (compute) begin

  case (i)

          22 : begin
                                        temp_num[(25*22) - 1 : 25*(22 - 1)] <=
(temp_num[(25*(22 + 1)) - 1 : 25*22] >= {temp_root[(23*(22 + 1)) - 1 : 23*22], 2'b01})
?
                {(temp_num[(25*(22 + 1)) - 3 : 25*22] - {temp_root[(23*(22 + 1)) - 3 :
23*22], 2'b01}), x[(2*22) - 1 : (2*22) - 2]} :
                {temp_num[(25*(22 + 1)) - 3 : 25*22], x[(2*22) - 1 : (2*22) - 2]};

                                        temp_root[(23*22) - 1 : 23*(22 - 1)] <=
(temp_num[(25*(22 + 1)) - 1 : 25*22] >= {temp_root[(23*(22 + 1)) - 1 : 23*22], 2'b01})
?
                {temp_root[(23*(22 + 1)) - 2 : 23*22], 1'b1} :
                {temp_root[(23*(22 + 1)) - 2 : 23*22], 1'b0};

                        i <= i - 1;
                                end

          21 : begin
```

```
                                        temp_num[(25*21) - 1 : 25*(21 - 1)] <=
(temp_num[(25*(21 + 1)) - 1 : 25*21] >= {temp_root[(23*(21 + 1)) - 1 : 23*21], 2'b01})
?
                {(temp_num[(25*(21 + 1)) - 3 : 25*21] - {temp_root[(23*(21 + 1)) - 3 :
23*21], 2'b01}), x[(2*21) - 1 : (2*21) - 2]} :
                {temp_num[(25*(21 + 1)) - 3 : 25*21], x[(2*21) - 1 : (2*21) - 2]};

                                        temp_root[(23*21) - 1 : 23*(21 - 1)] <=
(temp_num[(25*(21 + 1)) - 1 : 25*21] >= {temp_root[(23*(21 + 1)) - 1 : 23*21], 2'b01})
?
                {temp_root[(23*(21 + 1)) - 2 : 23*21], 1'b1} :
                {temp_root[(23*(21 + 1)) - 2 : 23*21], 1'b0};

                                    i <= i - 1;
                                        end
            20 : begin
                                        temp_num[(25*20) - 1 : 25*(20 - 1)] <=
(temp_num[(25*(20 + 1)) - 1 : 25*20] >= {temp_root[(23*(20 + 1)) - 1 : 23*20], 2'b01})
?
                {(temp_num[(25*(20 + 1)) - 3 : 25*20] - {temp_root[(23*(20 + 1)) - 3 :
23*20], 2'b01}), x[(2*20) - 1 : (2*20) - 2]} :
                {temp_num[(25*(20 + 1)) - 3 : 25*20], x[(2*20) - 1 : (2*20) - 2]};

                                        temp_root[(23*20) - 1 : 23*(20 - 1)] <=
(temp_num[(25*(20 + 1)) - 1 : 25*20] >= {temp_root[(23*(20 + 1)) - 1 : 23*20], 2'b01})
?
                {temp_root[(23*(20 + 1)) - 2 : 23*20], 1'b1} :
                {temp_root[(23*(20 + 1)) - 2 : 23*20], 1'b0};

                                        i <= i - 1;
                                        end

            19 : begin
                                        temp_num[(25*19) - 1 : 25*(19 - 1)] <=
(temp_num[(25*(19 + 1)) - 1 : 25*19] >= {temp_root[(23*(19 + 1)) - 1 : 23*19], 2'b01})
?
                {(temp_num[(25*(19 + 1)) - 3 : 25*19] - {temp_root[(23*(19 + 1)) - 3 :
23*19], 2'b01}), x[(2*19) - 1 : (2*19) - 2]} :
                {temp_num[(25*(19 + 1)) - 3 : 25*19], x[(2*19) - 1 : (2*19) - 2]};

                                        temp_root[(23*19) - 1 : 23*(19 - 1)] <=
(temp_num[(25*(19 + 1)) - 1 : 25*19] >= {temp_root[(23*(19 + 1)) - 1 : 23*19], 2'b01})
?
                {temp_root[(23*(19 + 1)) - 2 : 23*19], 1'b1} :
                {temp_root[(23*(19 + 1)) - 2 : 23*19], 1'b0};
```

```
                                        i <= i - 1;
                                        end

        18 : begin
                                        temp_num[(25*18) - 1 : 25*(18 - 1)] <=
(temp_num[(25*(18 + 1)) - 1 : 25*18] >= {temp_root[(23*(18 + 1)) - 1 : 23*18], 2'b01})
?
                {(temp_num[(25*(18 + 1)) - 3 : 25*18] - {temp_root[(23*(18 + 1)) - 3 :
23*18], 2'b01}), x[(2*18) - 1 : (2*18) - 2]} :
                {temp_num[(25*(18 + 1)) - 3 : 25*18], x[(2*18) - 1 : (2*18) - 2]};

                                        temp_root[(23*18) - 1 : 23*(18 - 1)] <=
(temp_num[(25*(18 + 1)) - 1 : 25*18] >= {temp_root[(23*(18 + 1)) - 1 : 23*18], 2'b01})
?
                {temp_root[(23*(18 + 1)) - 2 : 23*18], 1'b1} :
                {temp_root[(23*(18 + 1)) - 2 : 23*18], 1'b0};

                                        i <= i - 1;
                                        end

        17 : begin
                                        temp_num[(25*17) - 1 : 25*(17 - 1)] <=
(temp_num[(25*(17 + 1)) - 1 : 25*17] >= {temp_root[(23*(17 + 1)) - 1 : 23*17], 2'b01})
?
                {(temp_num[(25*(17 + 1)) - 3 : 25*17] - {temp_root[(23*(17 + 1)) - 3 :
23*17], 2'b01}), x[(2*17) - 1 : (2*17) - 2]} :
                {temp_num[(25*(17 + 1)) - 3 : 25*17], x[(2*17) - 1 : (2*17) - 2]};

                                        temp_root[(23*17) - 1 : 23*(17 - 1)] <=
(temp_num[(25*(17 + 1)) - 1 : 25*17] >= {temp_root[(23*(17 + 1)) - 1 : 23*17], 2'b01})
?
                {temp_root[(23*(17 + 1)) - 2 : 23*17], 1'b1} :
                {temp_root[(23*(17 + 1)) - 2 : 23*17], 1'b0};

                                        i <= i - 1;
                                        end

        16 : begin
                                        temp_num[(25*16) - 1 : 25*(16 - 1)] <=
(temp_num[(25*(16 + 1)) - 1 : 25*16] >= {temp_root[(23*(16 + 1)) - 1 : 23*16], 2'b01})
?
                {(temp_num[(25*(16 + 1)) - 3 : 25*16] - {temp_root[(23*(16 + 1)) - 3 :
23*16], 2'b01}), x[(2*16) - 1 : (2*16) - 2]} :
                {temp_num[(25*(16 + 1)) - 3 : 25*16], x[(2*16) - 1 : (2*16) - 2]};
```

```
                                    temp_root[(23*16) - 1 : 23*(16 - 1)] <=
(temp_num[(25*(16 + 1)) - 1 : 25*16] >= {temp_root[(23*(16 + 1)) - 1 : 23*16], 2'b01})
?
                {temp_root[(23*(16 + 1)) - 2 : 23*16], 1'b1} :
                {temp_root[(23*(16 + 1)) - 2 : 23*16], 1'b0};

                                    i <= i - 1;
                                    end

        15 : begin
                                    temp_num[(25*15) - 1 : 25*(15 - 1)] <=
(temp_num[(25*(15 + 1)) - 1 : 25*15] >= {temp_root[(23*(15 + 1)) - 1 : 23*15], 2'b01})
?
                {(temp_num[(25*(15 + 1)) - 3 : 25*15] - {temp_root[(23*(15 + 1)) - 3 :
23*15], 2'b01}), x[(2*15) - 1 : (2*15) - 2]} :
                {temp_num[(25*(15 + 1)) - 3 : 25*15], x[(2*15) - 1 : (2*15) - 2]};

                                    temp_root[(23*15) - 1 : 23*(15 - 1)] <=
(temp_num[(25*(15 + 1)) - 1 : 25*15] >= {temp_root[(23*(15 + 1)) - 1 : 23*15], 2'b01})
?
                {temp_root[(23*(15 + 1)) - 2 : 23*15], 1'b1} :
                {temp_root[(23*(15 + 1)) - 2 : 23*15], 1'b0};

                                    i <= i - 1;
                                    end

        14 : begin
                                    temp_num[(25*14) - 1 : 25*(14 - 1)] <=
(temp_num[(25*(14 + 1)) - 1 : 25*14] >= {temp_root[(23*(14 + 1)) - 1 : 23*14], 2'b01})
?
                {(temp_num[(25*(14 + 1)) - 3 : 25*14] - {temp_root[(23*(14 + 1)) - 3 :
23*14], 2'b01}), x[(2*14) - 1 : (2*14) - 2]} :
                {temp_num[(25*(14 + 1)) - 3 : 25*14], x[(2*14) - 1 : (2*14) - 2]};

                                    temp_root[(23*14) - 1 : 23*(14 - 1)] <=
(temp_num[(25*(14 + 1)) - 1 : 25*14] >= {temp_root[(23*(14 + 1)) - 1 : 23*14], 2'b01})
?
                {temp_root[(23*(14 + 1)) - 2 : 23*14], 1'b1} :
                {temp_root[(23*(14 + 1)) - 2 : 23*14], 1'b0};

                                    i <= i - 1;
                                    end

        13 : begin
```

```
                                        temp_num[(25*13) - 1 : 25*(13 - 1)] <=
(temp_num[(25*(13 + 1)) - 1 : 25*13] >= {temp_root[(23*(13 + 1)) - 1 : 23*13], 2'b01})
?
                {(temp_num[(25*(13 + 1)) - 3 : 25*13] - {temp_root[(23*(13 + 1)) - 3 :
23*13], 2'b01}), x[(2*13) - 1 : (2*13) - 2]} :
                {temp_num[(25*(13 + 1)) - 3 : 25*13], x[(2*13) - 1 : (2*13) - 2]};

                                        temp_root[(23*13) - 1 : 23*(13 - 1)] <=
(temp_num[(25*(13 + 1)) - 1 : 25*13] >= {temp_root[(23*(13 + 1)) - 1 : 23*13], 2'b01})
?
                {temp_root[(23*(13 + 1)) - 2 : 23*13], 1'b1} :
                {temp_root[(23*(13 + 1)) - 2 : 23*13], 1'b0};

                                        i <= i - 1;
                                        end

        12 : begin
                                        temp_num[(25*12) - 1 : 25*(12 - 1)] <=
(temp_num[(25*(12 + 1)) - 1 : 25*12] >= {temp_root[(23*(12 + 1)) - 1 : 23*12], 2'b01})
?
                {(temp_num[(25*(12 + 1)) - 3 : 25*12] - {temp_root[(23*(12 + 1)) - 3 :
23*12], 2'b01}), x[(2*12) - 1 : (2*12) - 2]} :
                {temp_num[(25*(12 + 1)) - 3 : 25*12], x[(2*12) - 1 : (2*12) - 2]};

                                        temp_root[(23*12) - 1 : 23*(12 - 1)] <=
(temp_num[(25*(12 + 1)) - 1 : 25*12] >= {temp_root[(23*(12 + 1)) - 1 : 23*12], 2'b01})
?
                {temp_root[(23*(12 + 1)) - 2 : 23*12], 1'b1} :
                {temp_root[(23*(12 + 1)) - 2 : 23*12], 1'b0};

                                        i <= i - 1;
                                        end

        11 : begin
                                        temp_num[(25*11) - 1 : 25*(11 - 1)] <=
(temp_num[(25*(11 + 1)) - 1 : 25*11] >= {temp_root[(23*(11 + 1)) - 1 : 23*11], 2'b01})
?
                {(temp_num[(25*(11 + 1)) - 3 : 25*11] - {temp_root[(23*(11 + 1)) - 3 :
23*11], 2'b01}), x[(2*11) - 1 : (2*11) - 2]} :
                {temp_num[(25*(11 + 1)) - 3 : 25*11], x[(2*11) - 1 : (2*11) - 2]};

                                        temp_root[(23*11) - 1 : 23*(11 - 1)] <=
(temp_num[(25*(11 + 1)) - 1 : 25*11] >= {temp_root[(23*(11 + 1)) - 1 : 23*11], 2'b01})
?
                {temp_root[(23*(11 + 1)) - 2 : 23*11], 1'b1} :
                {temp_root[(23*(11 + 1)) - 2 : 23*11], 1'b0};
```

```
                                        i <= i - 1;
                                        end

        10 : begin
                                        temp_num[(25*10) - 1 : 25*(10 - 1)] <=
(temp_num[(25*(10 + 1)) - 1 : 25*10] >= {temp_root[(23*(10 + 1)) - 1 : 23*10], 2'b01})
?
                {(temp_num[(25*(10 + 1)) - 3 : 25*10] - {temp_root[(23*(10 + 1)) - 3 :
23*10], 2'b01}), x[(2*10) - 1 : (2*10) - 2]} :
                {temp_num[(25*(10 + 1)) - 3 : 25*10], x[(2*10) - 1 : (2*10) - 2]};

                                        temp_root[(23*10) - 1 : 23*(10 - 1)] <=
(temp_num[(25*(10 + 1)) - 1 : 25*10] >= {temp_root[(23*(10 + 1)) - 1 : 23*10], 2'b01})
?
                {temp_root[(23*(10 + 1)) - 2 : 23*10], 1'b1} :
                {temp_root[(23*(10 + 1)) - 2 : 23*10], 1'b0};

                                        i <= i - 1;
                                        end

        9 : begin
                                        temp_num[(25*9) - 1 : 25*(9 - 1)] <=
(temp_num[(25*(9 + 1)) - 1 : 25*9] >= {temp_root[(23*(9 + 1)) - 1 : 23*9], 2'b01}) ?
                {(temp_num[(25*(9 + 1)) - 3 : 25*9] - {temp_root[(23*(9 + 1)) - 3 : 23*9],
2'b01}), x[(2*9) - 1 : (2*9) - 2]} :
                {temp_num[(25*(9 + 1)) - 3 : 25*9], x[(2*9) - 1 : (2*9) - 2]};

                                        temp_root[(23*9) - 1 : 23*(9 - 1)] <=
(temp_num[(25*(9 + 1)) - 1 : 25*9] >= {temp_root[(23*(9 + 1)) - 1 : 23*9], 2'b01}) ?
                {temp_root[(23*(9 + 1)) - 2 : 23*9], 1'b1} :
                {temp_root[(23*(9 + 1)) - 2 : 23*9], 1'b0};

                                        i <= i - 1;
                                        end

        8 : begin
                                        temp_num[(25*8) - 1 : 25*(8 - 1)] <=
(temp_num[(25*(8 + 1)) - 1 : 25*8] >= {temp_root[(23*(8 + 1)) - 1 : 23*8], 2'b01}) ?
                {(temp_num[(25*(8 + 1)) - 3 : 25*8] - {temp_root[(23*(8 + 1)) - 3 : 23*8],
2'b01}), x[(2*8) - 1 : (2*8) - 2]} :
                {temp_num[(25*(8 + 1)) - 3 : 25*8], x[(2*8) - 1 : (2*8) - 2]};

                                        temp_root[(23*8) - 1 : 23*(8 - 1)] <=
(temp_num[(25*(8 + 1)) - 1 : 25*8] >= {temp_root[(23*(8 + 1)) - 1 : 23*8], 2'b01}) ?
                {temp_root[(23*(8 + 1)) - 2 : 23*8], 1'b1} :
```

```
                {temp_root[(23*(8 + 1)) - 2 : 23*8], 1'b0};

                                        i <= i - 1;
                                        end

        7 : begin
                                        temp_num[(25*7) - 1 : 25*(7 - 1)] <=
(temp_num[(25*(7 + 1)) - 1 : 25*7] >= {temp_root[(23*(7 + 1)) - 1 : 23*7], 2'b01}) ?
                {(temp_num[(25*(7 + 1)) - 3 : 25*7] - {temp_root[(23*(7 + 1)) - 3 : 23*7],
2'b01}), x[(2*7) - 1 : (2*7) - 2]} :
                {temp_num[(25*(7 + 1)) - 3 : 25*7], x[(2*7) - 1 : (2*7) - 2]};

                                        temp_root[(23*7) - 1 : 23*(7 - 1)] <=
(temp_num[(25*(7 + 1)) - 1 : 25*7] >= {temp_root[(23*(7 + 1)) - 1 : 23*7], 2'b01}) ?
                {temp_root[(23*(7 + 1)) - 2 : 23*7], 1'b1} :
                {temp_root[(23*(7 + 1)) - 2 : 23*7], 1'b0};
                                        i <= i - 1;
                                        end

        6 : begin
                                        temp_num[(25*6) - 1 : 25*(6 - 1)] <=
(temp_num[(25*(6 + 1)) - 1 : 25*6] >= {temp_root[(23*(6 + 1)) - 1 : 23*6], 2'b01}) ?
                {(temp_num[(25*(6 + 1)) - 3 : 25*6] - {temp_root[(23*(6 + 1)) - 3 : 23*6],
2'b01}), x[(2*6) - 1 : (2*6) - 2]} :
                {temp_num[(25*(6 + 1)) - 3 : 25*6], x[(2*6) - 1 : (2*6) - 2]};

                                        temp_root[(23*6) - 1 : 23*(6 - 1)] <=
(temp_num[(25*(6 + 1)) - 1 : 25*6] >= {temp_root[(23*(6 + 1)) - 1 : 23*6], 2'b01}) ?
                {temp_root[(23*(6 + 1)) - 2 : 23*6], 1'b1} :
                {temp_root[(23*(6 + 1)) - 2 : 23*6], 1'b0};
                                        i <= i - 1;
                                        end

        5 : begin
                                        temp_num[(25*5) - 1 : 25*(5 - 1)] <=
(temp_num[(25*(5 + 1)) - 1 : 25*5] >= {temp_root[(23*(5 + 1)) - 1 : 23*5], 2'b01}) ?
                {(temp_num[(25*(5 + 1)) - 3 : 25*5] - {temp_root[(23*(5 + 1)) - 3 : 23*5],
2'b01}), x[(2*5) - 1 : (2*5) - 2]} :
                {temp_num[(25*(5 + 1)) - 3 : 25*5], x[(2*5) - 1 : (2*5) - 2]};

                                        temp_root[(23*5) - 1 : 23*(5 - 1)] <=
(temp_num[(25*(5 + 1)) - 1 : 25*5] >= {temp_root[(23*(5 + 1)) - 1 : 23*5], 2'b01}) ?
                {temp_root[(23*(5 + 1)) - 2 : 23*5], 1'b1} :
                {temp_root[(23*(5 + 1)) - 2 : 23*5], 1'b0};
                                        i <= i - 1;
                                        end
```

```
        4 : begin
                                        temp_num[(25*4) - 1 : 25*(4 - 1)] <=
(temp_num[(25*(4 + 1)) - 1 : 25*4] >= {temp_root[(23*(4 + 1)) - 1 : 23*4], 2'b01}) ?
                {(temp_num[(25*(4 + 1)) - 3 : 25*4] - {temp_root[(23*(4 + 1)) - 3 : 23*4],
2'b01}), x[(2*4) - 1 : (2*4) - 2]} :
                {temp_num[(25*(4 + 1)) - 3 : 25*4], x[(2*4) - 1 : (2*4) - 2]};

                                        temp_root[(23*4) - 1 : 23*(4 - 1)] <=
(temp_num[(25*(4 + 1)) - 1 : 25*4] >= {temp_root[(23*(4 + 1)) - 1 : 23*4], 2'b01}) ?
                {temp_root[(23*(4 + 1)) - 2 : 23*4], 1'b1} :
                {temp_root[(23*(4 + 1)) - 2 : 23*4], 1'b0};
                                        i <= i - 1;
                                        end

        3 : begin
                                        temp_num[(25*3) - 1 : 25*(3 - 1)] <=
(temp_num[(25*(3 + 1)) - 1 : 25*3] >= {temp_root[(23*(3 + 1)) - 1 : 23*3], 2'b01}) ?
                {(temp_num[(25*(3 + 1)) - 3 : 25*3] - {temp_root[(23*(3 + 1)) - 3 : 23*3],
2'b01}), x[(2*3) - 1 : (2*3) - 2]} :
                {temp_num[(25*(3 + 1)) - 3 : 25*3], x[(2*3) - 1 : (2*3) - 2]};

                                        temp_root[(23*3) - 1 : 23*(3 - 1)] <=
(temp_num[(25*(3 + 1)) - 1 : 25*3] >= {temp_root[(23*(3 + 1)) - 1 : 23*3], 2'b01}) ?
                {temp_root[(23*(3 + 1)) - 2 : 23*3], 1'b1} :
                {temp_root[(23*(3 + 1)) - 2 : 23*3], 1'b0};
                                        i <= i - 1;
                                        end

        2 : begin
                                        temp_num[(25*2) - 1 : 25*(2 - 1)] <=
(temp_num[(25*(2 + 1)) - 1 : 25*2] >= {temp_root[(23*(2 + 1)) - 1 : 23*2], 2'b01}) ?
                {(temp_num[(25*(2 + 1)) - 3 : 25*2] - {temp_root[(23*(2 + 1)) - 3 : 23*2],
2'b01}), x[(2*2) - 1 : (2*2) - 2]} :
                {temp_num[(25*(2 + 1)) - 3 : 25*2], x[(2*2) - 1 : (2*2) - 2]};

                                        temp_root[(23*2) - 1 : 23*(2 - 1)] <=
(temp_num[(25*(2 + 1)) - 1 : 25*2] >= {temp_root[(23*(2 + 1)) - 1 : 23*2], 2'b01}) ?
                {temp_root[(23*(2 + 1)) - 2 : 23*2], 1'b1} :
                {temp_root[(23*(2 + 1)) - 2 : 23*2], 1'b0};
                                        i <= i - 1;
                                        end

        1 : begin
                                        temp_num[(25*1) - 1 : 25*(1 - 1)] <=
(temp_num[(25*(1 + 1)) - 1 : 25*1] >= {temp_root[(23*(1 + 1)) - 3 : 23*1], 2'b01}) ?
```

```
                {(temp_num[(25*(1 + 1)) - 3 : 25*1] - {temp_root[(23*(1 + 1)) - 3 : 23*1],
2'b01}), x[(2*1) - 1 : (2*1) - 2]} :
                {temp_num[(25*(1 + 1)) - 3 : 25*1], x[(2*1) - 1 : (2*1) - 2]};

                                        temp_root[(23*1) - 1 : 23*(1 - 1)] <=
(temp_num[(25*(1 + 1)) - 1 : 25*1] >= {temp_root[(23*(1 + 1)) - 3 : 23*1], 2'b01}) ?
                {temp_root[(23*(1 + 1)) - 2 : 23*1], 1'b1} :
                {temp_root[(23*(1 + 1)) - 2 : 23*1], 1'b0};

                                        i <= i - 1;
                                        end

            default : begin
                                                sqrt_x <= (temp_num[24:0] >=
{temp_root[22:0], 2'b01}) ?
                {temp_root[21:0], 1'b1}:
                {temp_root[21:0], 1'b0};
                                                compute <= 0;
                                                done <= 1;
                                                i <= 22;
                                                end

            endcase

            end
```

```
//   if (i == 0) begin
//      sqrt_x <= (temp_num[24:0] >= {(temp_root[22:0] << 2), 2'b01}) ?
//                {(temp_root[22:0] << 1), 1'b1}:
//                {(temp_root[22:0] << 1), 1'b0};
//      compute <= 0;
//      done <= 1;
//      i <= 22;
//      end
//   else begin
//      temp_num[(25*i) - 1 : 25*(i - 1)] <= (temp_num[(25*(i + 1)) - 1 : 25*i] >=
{(temp_root[(23*(i + 1)) - 1 : 23*i] << 2), 2'b01}) ?
//                   {((temp_num[(25*(i + 1)) - 1 : 25*i] - {(temp_root[(23*(i + 1)) - 1 : 23*i]
<< 2), 2'b01}) << 2), x[(2*i) - 1 : (2*i) - 2]} :
//                   {(temp_num[(25*(i + 1)) - 1 : 25*i] << 2), x[(2*i) - 1 : (2*i) - 2]};
```

```
//
//      temp_root[(23*i) - 1 : 23*(i - 1)] <= (temp_num[(25*(i + 1)) - 1 : 25*i] >=
{(temp_root[(23*(i + 1)) - 1 : 23*i] << 2), 2'b01}) ?
//                      {(temp_root[(23*(i + 1)) - 1 : 23*i] << 1), 1'b1} :
//                      {(temp_root[(23*(i + 1)) - 1 : 23*i] << 1), 1'b0};
//
//      i <= i - 1;
//      end
//   end

else begin
   done <= 0;
   sqrt_x <= sqrt_x;
   end

end


endmodule
```