```
module DE1_SoC_Computer (
	////////////////////////////////////
	// FPGA Pins
	////////////////////////////////////

	// Clock pins
	CLOCK_50,
	CLOCK2_50,
	CLOCK3_50,
	CLOCK4_50,

	// ADC
	ADC_CS_N,
	ADC_DIN,
	ADC_DOUT,
	ADC_SCLK,

	// Audio
	AUD_ADCDAT,
	AUD_ADCLRCK,
	AUD_BCLK,
	AUD_DACDAT,
	AUD_DACLRCK,
	AUD_XCK,

	// SDRAM
	DRAM_ADDR,
	DRAM_BA,
	DRAM_CAS_N,
	DRAM_CKE,
	DRAM_CLK,
	DRAM_CS_N,
	DRAM_DQ,
	DRAM_LDQM,
	DRAM_RAS_N,
	DRAM_UDQM,
	DRAM_WE_N,

	// I2C Bus for Configuration of the Audio and Video-In Chips
	FPGA_I2C_SCLK,
	FPGA_I2C_SDAT,
```

```
// 40-Pin Headers
GPIO_0,
GPIO_1,

// Seven Segment Displays
HEX0,
HEX1,
HEX2,
HEX3,
HEX4,
HEX5,

// IR
IRDA_RXD,
IRDA_TXD,

// Pushbuttons
KEY,

// LEDs
LEDR,

// PS2 Ports
PS2_CLK,
PS2_DAT,

PS2_CLK2,
PS2_DAT2,

// Slider Switches
SW,

// Video-In
TD_CLK27,
TD_DATA,
TD_HS,
TD_RESET_N,
TD_VS,

// VGA
VGA_B,
VGA_BLANK_N,
VGA_CLK,
VGA_G,
```

```
VGA_HS,
VGA_R,
VGA_SYNC_N,
VGA_VS,

///////////////////////////////////
// HPS Pins
///////////////////////////////////

// DDR3 SDRAM
HPS_DDR3_ADDR,
HPS_DDR3_BA,
HPS_DDR3_CAS_N,
HPS_DDR3_CKE,
HPS_DDR3_CK_N,
HPS_DDR3_CK_P,
HPS_DDR3_CS_N,
HPS_DDR3_DM,
HPS_DDR3_DQ,
HPS_DDR3_DQS_N,
HPS_DDR3_DQS_P,
HPS_DDR3_ODT,
HPS_DDR3_RAS_N,
HPS_DDR3_RESET_N,
HPS_DDR3_RZQ,
HPS_DDR3_WE_N,

// Ethernet
HPS_ENET_GTX_CLK,
HPS_ENET_INT_N,
HPS_ENET_MDC,
HPS_ENET_MDIO,
HPS_ENET_RX_CLK,
HPS_ENET_RX_DATA,
HPS_ENET_RX_DV,
HPS_ENET_TX_DATA,
HPS_ENET_TX_EN,

// Flash
HPS_FLASH_DATA,
HPS_FLASH_DCLK,
HPS_FLASH_NCSO,

// Accelerometer
```

```
        HPS_GSENSOR_INT,

        // General Purpose I/O
        HPS_GPIO,

        // I2C
        HPS_I2C_CONTROL,
        HPS_I2C1_SCLK,
        HPS_I2C1_SDAT,
        HPS_I2C2_SCLK,
        HPS_I2C2_SDAT,

        // Pushbutton
        HPS_KEY,

        // LED
        HPS_LED,

        // SD Card
        HPS_SD_CLK,
        HPS_SD_CMD,
        HPS_SD_DATA,

        // SPI
        HPS_SPIM_CLK,
        HPS_SPIM_MISO,
        HPS_SPIM_MOSI,
        HPS_SPIM_SS,

        // UART
        HPS_UART_RX,
        HPS_UART_TX,

        // USB
        HPS_CONV_USB_N,
        HPS_USB_CLKOUT,
        HPS_USB_DATA,
        HPS_USB_DIR,
        HPS_USB_NXT,
        HPS_USB_STP
);

//=======================================================
//  PARAMETER declarations
```

```
//=======================================================


//=======================================================
//  PORT declarations
//=======================================================

///////////////////////////////////
// FPGA Pins
///////////////////////////////////

// Clock pins
input                                   CLOCK_50;
input                                   CLOCK2_50;
input                                   CLOCK3_50;
input                                   CLOCK4_50;

// ADC
inout                                   ADC_CS_N;
output                                  ADC_DIN;
input                                   ADC_DOUT;
output                                  ADC_SCLK;

// Audio
input                                   AUD_ADCDAT;
inout                                   AUD_ADCLRCK;
inout                                   AUD_BCLK;
output                                  AUD_DACDAT;
inout                                   AUD_DACLRCK;
output                                  AUD_XCK;

// SDRAM
output          [12: 0]  DRAM_ADDR;
output          [ 1: 0]  DRAM_BA;
output                                  DRAM_CAS_N;
output                                  DRAM_CKE;
output                                  DRAM_CLK;
output                                  DRAM_CS_N;
inout                   [15: 0]  DRAM_DQ;
output                                  DRAM_LDQM;
output                                  DRAM_RAS_N;
output                                  DRAM_UDQM;
output                                  DRAM_WE_N;
```

```
// I2C Bus for Configuration of the Audio and Video-In Chips
output                  FPGA_I2C_SCLK;
inout                   FPGA_I2C_SDAT;

// 40-pin headers
inout       [35: 0]  GPIO_0;
inout       [35: 0]  GPIO_1;

// Seven Segment Displays
output      [ 6: 0]  HEX0;
output      [ 6: 0]  HEX1;
output      [ 6: 0]  HEX2;
output      [ 6: 0]  HEX3;
output      [ 6: 0]  HEX4;
output      [ 6: 0]  HEX5;

// IR
input                   IRDA_RXD;
output                  IRDA_TXD;

// Pushbuttons
input       [ 3: 0]  KEY;

// LEDs
output      [ 9: 0]  LEDR;

// PS2 Ports
inout                   PS2_CLK;
inout                   PS2_DAT;

inout                   PS2_CLK2;
inout                   PS2_DAT2;

// Slider Switches
input       [ 9: 0]  SW;

// Video-In
input                   TD_CLK27;
input       [ 7: 0]  TD_DATA;
input                   TD_HS;
output                  TD_RESET_N;
input                   TD_VS;

// VGA
```

```
output          [ 7: 0]  VGA_B;
output                   VGA_BLANK_N;
output                   VGA_CLK;
output          [ 7: 0]  VGA_G;
output                   VGA_HS;
output          [ 7: 0]  VGA_R;
output                   VGA_SYNC_N;
output                   VGA_VS;



///////////////////////////////////
// HPS Pins
///////////////////////////////////

// DDR3 SDRAM
output          [14: 0]  HPS_DDR3_ADDR;
output          [ 2: 0]  HPS_DDR3_BA;
output                   HPS_DDR3_CAS_N;
output                   HPS_DDR3_CKE;
output                   HPS_DDR3_CK_N;
output                   HPS_DDR3_CK_P;
output                   HPS_DDR3_CS_N;
output          [ 3: 0]  HPS_DDR3_DM;
inout           [31: 0]  HPS_DDR3_DQ;
inout           [ 3: 0]  HPS_DDR3_DQS_N;
inout           [ 3: 0]  HPS_DDR3_DQS_P;
output                   HPS_DDR3_ODT;
output                   HPS_DDR3_RAS_N;
output                   HPS_DDR3_RESET_N;
input                    HPS_DDR3_RZQ;
output                   HPS_DDR3_WE_N;

// Ethernet
output                   HPS_ENET_GTX_CLK;
inout                    HPS_ENET_INT_N;
output                   HPS_ENET_MDC;
inout                    HPS_ENET_MDIO;
input                    HPS_ENET_RX_CLK;
input           [ 3: 0]  HPS_ENET_RX_DATA;
input                    HPS_ENET_RX_DV;
output          [ 3: 0]  HPS_ENET_TX_DATA;
output                   HPS_ENET_TX_EN;
```

```
// Flash
inout       [ 3: 0]  HPS_FLASH_DATA;
output               HPS_FLASH_DCLK;
output               HPS_FLASH_NCSO;

// Accelerometer
inout                HPS_GSENSOR_INT;

// General Purpose I/O
inout       [ 1: 0]  HPS_GPIO;

// I2C
inout                HPS_I2C_CONTROL;
inout                HPS_I2C1_SCLK;
inout                HPS_I2C1_SDAT;
inout                HPS_I2C2_SCLK;
inout                HPS_I2C2_SDAT;

// Pushbutton
inout                HPS_KEY;

// LED
inout                HPS_LED;

// SD Card
output               HPS_SD_CLK;
inout                HPS_SD_CMD;
inout       [ 3: 0]  HPS_SD_DATA;

// SPI
output               HPS_SPIM_CLK;
input                HPS_SPIM_MISO;
output               HPS_SPIM_MOSI;
inout                HPS_SPIM_SS;

// UART
input                HPS_UART_RX;
output               HPS_UART_TX;

// USB
inout                HPS_CONV_USB_N;
input                HPS_USB_CLKOUT;
inout       [ 7: 0]  HPS_USB_DATA;
input                HPS_USB_DIR;
```

```
input                                           HPS_USB_NXT;
output                                      HPS_USB_STP;

//=======================================================
//  REG/WIRE declarations
//=======================================================

wire                [15: 0]  hex3_hex0;
//wire                [15: 0]  hex5_hex4;

//assign HEX0 = ~hex3_hex0[ 6: 0]; // hex3_hex0[ 6: 0];
//assign HEX1 = ~hex3_hex0[14: 8];
//assign HEX2 = ~hex3_hex0[22:16];
//assign HEX3 = ~hex3_hex0[30:24];
assign HEX4 = 7'b1111111;
assign HEX5 = 7'b1111111;

HexDigit Digit0(HEX0, hex3_hex0[3:0]);
HexDigit Digit1(HEX1, hex3_hex0[7:4]);
HexDigit Digit2(HEX2, hex3_hex0[11:8]);
HexDigit Digit3(HEX3, hex3_hex0[15:12]);

//=======================================================
// SRAM/VGA state machine
//=======================================================
// --Check for sram address=0 nonzero, which means that
//   HPS wrote some new data.
//
// --Read sram address 1 and 2 to get x1, y1
//   left-most x, upper-most y
// --Read sram address 3 and 4 to get x2, y2
//   right-most x, lower-most y
// --Read sram address 5 to get color
// --write a rectangle to VGA
//
// --clear sram address=0 to signal HPS
//=======================================================
// Controls for Qsys sram slave exported in system module
//=======================================================
wire [31:0] sram_readdata ;
reg [31:0] data_buffer, sram_writedata ;
reg [10:0] sram_address;
reg sram_write;
wire sram_clken = 1'b1;
```

```
wire sram_chipselect = 1'b1;
//reg [7:0] state;

// rectangle corners
//reg [9:0] x1, y1, x2, y2 ;
reg [31:0] timer ; // may need to throttle write-rate
//=======================================================
// Controls for VGA memory
//=======================================================
wire [31:0] vga_out_base_address = 32'h0000_0000 ;  // vga base addr
reg [7:0] vga_sram_writedata ;
reg [31:0] vga_sram_address;
reg vga_sram_write ;
wire vga_sram_clken = 1'b1;
wire vga_sram_chipselect = 1'b1;
reg [9:0] vga_x_cood, vga_y_cood;

//=======================================================


//conv2d genius(.clk(CLOCK_50),.reset(pio_rst),.result(result));

//=======================================================
        parameter wsize=3;
        parameter isize=30;
        //we use 6.21 floating representation
        //for each 27*27 muliplier it needs one DSP block
        //input  clk,reset;
        //output [26:0] result;   // Ix

        reg  signed [26:0] d;
        wire signed [26:0] d_Ix2, d_Iy2, d_Ixy, d_R;
        wire signed [26:0] q, q_Ix2, q_Iy2, q_Ixy, q_R;
        reg        [9:0]  write_address,read_address;
        reg               we, we_Ix2, we_R;
        reg        [10:0] mem_addr;
        wire       [26:0] mem_out, mem_out_zero;

        reg signed  [26:0] kernel_Gx [0:wsize*wsize-1];
        reg signed  [26:0] kernel_Gy [0:wsize*wsize-1];
        reg signed  [26:0] kernel_Gxy [0:wsize*wsize-1];

        wire signed [26:0] result_x, result_y ,result_xy, result_Ix2, result_Iy2, result_Ixy;
```

```
        reg  signed [26:0] image_temp0 [0:wsize*wsize-1];
        reg  signed [26:0] image_temp1 [0:wsize*wsize-1];
        reg  signed [26:0] image_temp2 [0:wsize*wsize-1];

        wire signed [26:0] temp_x      [0:wsize*wsize-1];
        wire signed [26:0] temp_y      [0:wsize*wsize-1];
        wire signed [26:0] temp_xy     [0:wsize*wsize-1];

        wire signed [26:0] temp_add_x  [0:2];
        wire signed [26:0] temp_add_y  [0:2];
        wire signed [26:0] temp_add_xy [0:2];

        reg signed [26:0] kernel0    [0:wsize*wsize-1]; // kernel
        reg signed [26:0] kernel1    [0:wsize*wsize-1]; // kernel
        reg signed [26:0] kernel2    [0:wsize*wsize-1]; // kernel

        reg       [2:0]    x,y;
        reg       [3:0]    image_counter;
        wire signed [26:0] result_R;

        generate
                genvar n,m;
                for (n = 0; n < wsize*wsize; n = n + 1) begin: multGen
                        signed_mult multupliers_x( .a(kernel0[n]), .b(image_temp0[n]),
.out(temp_x[n]));
                        signed_mult multupliers_y( .a(kernel1[n]), .b(image_temp1[n]),
.out(temp_y[n]));
                        signed_mult multupliers_xy( .a(kernel2[n]), .b(image_temp2[n]),
.out(temp_xy[n])); // Sxy
                end

                for (m = 0; m <3 ;m=m+1)begin: addertree
                        adder adders_x(.a(temp_x[0+m*3]), .b(temp_x[1+m*3]),
.c(temp_x[2+m*3]), .out(temp_add_x[m]));
                        adder adders_y(.a(temp_y[0+m*3]), .b(temp_y[1+m*3]),
.c(temp_y[2+m*3]), .out(temp_add_y[m]));
                        adder adders_xy(.a(temp_xy[0+m*3]), .b(temp_xy[1+m*3]),
.c(temp_xy[2+m*3]), .out(temp_add_xy[m])); //Sxy
                end
        endgenerate

        adder adders0 (.a(temp_add_x[0]), .b(temp_add_x[1]), .c(temp_add_x[2]),
.out(result_x)); // Ix, Sx
```

```
        adder adders1 (.a(temp_add_y[0]), .b(temp_add_y[1]), .c(temp_add_y[2]),
.out(result_y)); // Iy, Sy
        adder adders2 (.a(temp_add_xy[0]), .b(temp_add_xy[1]), .c(temp_add_xy[2]),
.out(result_xy)); // Sxy

        signed_mult multupliers_Ix2( .a(result_x), .b(result_x), .out(result_Ix2)); // Ix2
        signed_mult multupliers_Iy2( .a(result_y), .b(result_y), .out(result_Iy2)); // Iy2
        signed_mult multupliers_Ixy( .a(result_x), .b(result_y), .out(result_Ixy)); // Ixy

        //sync_rom_image image_init (.clk(CLOCK_50),  .address(mem_addr),
.mem_out(mem_out));
        sync_rom_zero  zero_ped   (.clk(CLOCK_50),  .address(mem_addr),
.mem_out(mem_out_zero));
        //dual_clock_ram image      (.q(q), .d(mem_out), .write_address(write_address),
.read_address(read_address), .we(we), .clk1(CLOCK_50), .clk2(CLOCK_50));
        dual_clock_ram image      (.q(q), .d(sram_readdata[26:0]),
.write_address(write_address), .read_address(read_address), .we(we), .clk1(CLOCK_50),
.clk2(CLOCK_50));
        dual_clock_ram Ix2        (.q(q_Ix2), .d(d_Ix2), .write_address(write_address),
.read_address(read_address), .we(we_Ix2), .clk1(CLOCK_50), .clk2(CLOCK_50));
        dual_clock_ram Iy2        (.q(q_Iy2), .d(d_Iy2), .write_address(write_address),
.read_address(read_address), .we(we_Ix2), .clk1(CLOCK_50), .clk2(CLOCK_50));
        dual_clock_ram Ixy        (.q(q_Ixy), .d(d_Ixy), .write_address(write_address),
.read_address(read_address), .we(we_Ix2), .clk1(CLOCK_50), .clk2(CLOCK_50));
        dual_clock_ram R          (.q(q_R), .d(result_R), .write_address(write_address),
.read_address(read_address), .we(we_R), .clk1(CLOCK_50), .clk2(CLOCK_50));

        response tensor(.R(result_R), .x2(result_x), .y2(result_y), .xy(result_xy), .rst(pio_rst));


        parameter [5:0] state_signal_ready=6'd0, state_signal=6'd1, state_mem_prep = 6'd2,
state_mem_ready = 6'd3, state_mem =6'd4, state_der_ready = 6'd5, state_der = 6'd6,
state_der_idle = 6'd7, state_sum_ready = 6'd8, state_sum = 6'd9, state_idle = 6'd10;
        reg       [5:0] state;

        reg [4:0] row_counter, col_counter;
        integer i,j,k;

        reg [3:0] image_state;
        reg       addr_en;
        reg [4:0] read_counter;
        reg       isFirst, sram_en;
        reg [1:0] idle_counter;
        reg [9:0] tensor_counter;
```

```
assign d_lx2 = (state == state_mem) ? mem_out_zero : result_lx2;
assign d_ly2 = (state == state_mem) ? mem_out_zero : result_ly2;
assign d_lxy = (state == state_mem) ? mem_out_zero : result_lxy;

always @(posedge CLOCK_50) begin
	if (pio_rst) begin
		read_address <= 10'd0;
		read_counter <= 5'd0;
		image_state <= 4'd0;
	end else begin
		if (addr_en) begin
			if (image_state == 4'd0) begin
				read_address <= read_address + 1;
				image_state <= 4'd1;
			end

			if (image_state == 4'd1) begin
				read_address <= read_address + 1;
				image_state <= 4'd2;
			end

			if (image_state == 4'd2) begin
				read_address <= read_address + 30;
				image_state <= 4'd3;
			end

			if (image_state == 4'd3) begin
				read_address <= read_address + 1;
				image_state <= 4'd4;
			end

			if (image_state == 4'd4) begin
				read_address <= read_address + 1;
				image_state <= 4'd5;
			end

			if (image_state == 4'd5) begin
				read_address <= read_address + 30;
				image_state <= 4'd6;
			end

			if (image_state == 4'd6) begin
				read_address <= read_address + 1;
				image_state <= 4'd7;
```

```
                end

                if (image_state == 4'd7) begin
                        read_address <= read_address + 1;
                        image_state <= 4'd8;
                end

                if (image_state == 4'd8) begin
                        if (read_counter < 29) begin
                                read_address <= read_address - 65;
                                read_counter <= read_counter + 1;
                        end
                        else begin
                                read_address <= read_address - 63;
                                read_counter <= 0;
                        end
                        image_state <= 4'd0;
                end
        end else begin
                read_address <= 10'd0;
                read_counter <= 5'd0;
                image_state <= 4'd0;
        end
    end
end

always@(posedge CLOCK_50)begin
    if(pio_rst)begin
        //state<=state_mem;
        state <= state_signal_ready;

        mem_addr<=11'b0;
        write_address<=10'b0;
        //read_address <=10'b0;
        image_counter<=0;
        addr_en <= 1'b0;
        //image_state<=4'b0;
        isFirst    <=1'b1;

        //counters for indexing write Ix2 to M10K
        row_counter <= 5'b0;
        col_counter <= 5'b0;

        idle_counter <= 0;
```

```
tensor_counter <= 10'b0;

// vga
vga_sram_write <= 1'b0;
vga_x_cood <= 10'b0;
vga_y_cood <= 10'b0;

// hps handshake sram
sram_write <= 1'b0;
sram_en    <= 1'b0;

kernel_Gx[ 0 ] <= 27'b 111111101000011101001010100 ;
kernel_Gx[ 1 ] <= 27'b 000000000000000000000000000 ;
kernel_Gx[ 2 ] <= 27'b 000000101111100010110101011 ;
kernel_Gx[ 3 ] <= 27'b 111111011001001011101001101 ;
kernel_Gx[ 4 ] <= 27'b 000000000000000000000000000 ;
kernel_Gx[ 5 ] <= 27'b 000000100110110100010110010 ;
kernel_Gx[ 6 ] <= 27'b 111111101000011101001010100 ;
kernel_Gx[ 7 ] <= 27'b 000000000000000000000000000 ;
kernel_Gx[ 8 ] <= 27'b 000000101111100010110101011 ;

kernel_Gy[ 0 ] <= 27'b 111111101000011101001010100 ;
kernel_Gy[ 1 ] <= 27'b 111111011001001011101001101 ;
kernel_Gy[ 2 ] <= 27'b 111111101000011101001010100 ;
kernel_Gy[ 3 ] <= 27'b 000000000000000000000000000 ;
kernel_Gy[ 4 ] <= 27'b 000000000000000000000000000 ;
kernel_Gy[ 5 ] <= 27'b 000000000000000000000000000 ;
kernel_Gy[ 6 ] <= 27'b 000000101111100010110101011 ;
kernel_Gy[ 7 ] <= 27'b 000000100110110100010110010 ;
kernel_Gy[ 8 ] <= 27'b 000000101111100010110101011 ;

kernel_Gxy[ 0 ] <= 27'b 000000101111100010110101011 ;
kernel_Gxy[ 1 ] <= 27'b 000000100110110100010110010 ;
kernel_Gxy[ 2 ] <= 27'b 000000101111100010110101011 ;
kernel_Gxy[ 3 ] <= 27'b 000000100110110100010110010 ;
kernel_Gxy[ 4 ] <= 27'b 000001000000000000000000000 ;
kernel_Gxy[ 5 ] <= 27'b 000000100110110100010110010 ;
kernel_Gxy[ 6 ] <= 27'b 000000101111100010110101011 ;
kernel_Gxy[ 7 ] <= 27'b 000000100110110100010110010 ;
kernel_Gxy[ 8 ] <= 27'b 000000101111100010110101011 ;

//we <=1'b1; // one cycle ahead state mem
//we_lx2 <= 1'b1; // one cycle ahead state_mem
sram_address <= 11'd0;
```

```
		end
		else begin
			case(state)
				state_signal_ready: begin
					state <= state_signal;
				end

				state_signal: begin
					if (sram_readdata == 32'b1) begin
						state <= state_mem_prep;
					end else begin
						state <= state_signal; // polling
					end
				end

				state_mem_prep: begin
					sram_address <= 11'b1;
					sram_write <= 1'b0;
					state <= state_mem_ready;
				end

				state_mem_ready: begin
					we <=1'b1; // one cycle ahead state mem
					we_lx2 <= 1'b1; // one cycle ahead state_mem
					state <= state_mem;
					if (mem_addr == 11'd1024) begin
						addr_en <= 1'b1;	//use to generate the window temp read address pattern, fixed// make appear 3 clock ahead of the first read address update
					end
				end
				state_mem:begin
					if (mem_addr < 11'd1024)begin   //force to correct
						mem_addr     <= mem_addr+1'b1;
						//d           <= mem_out;
						write_address<= mem_addr;
						/*if (mem_addr == 11'd1023) begin
							addr_en <= 1'b1;	//use to generate the window temp read address pattern, fixed// make appear 3 clock ahead of the first read address update
						end*/
						sram_address <= sram_address + 1;
						state <= state_mem_ready;
					end else begin
						state <=state_der_ready;
```

```
                        we_Ix2 <= 1'b0;
                end
        end
        state_der_ready:begin
                we <= 1'b0;

                //we_Ix2 <= 1'b0;
                state<=state_der;
                for (i=0; i < wsize*wsize; i=i+1) begin
                        kernel0[i] <= kernel_Gx[i];
                        kernel1[i] <= kernel_Gy[i];
                end

        end
        state_der:begin
                if (col_counter < 5'd30) begin
                        we_Ix2 <= 1'b0; // write to M10K for every 8 cycles
                        image_temp0[image_counter] <= q;
                        image_temp1[image_counter] <= q;
                        //image_counter <= image_counter +1;
                        if (image_counter == 8) begin
                                if (isFirst)begin
                                        write_address <= 10'd33;
                                        isFirst <=1'b0;
                                end else begin
                                        if (row_counter < 5'd29) begin
                                                row_counter <= row_counter
+ 1;

                                                write_address <=
write_address + 1;
                                        end else begin
                                                row_counter <= 5'b0;
                                                col_counter <= col_counter +
1;

                                                write_address <=
write_address + 3;
                                        end
                                end
                                we_Ix2 <= 1'b1;
                                if (write_address == 10'd990) begin
                                        we_Ix2 <= 1'b0;
                                end
                                image_counter <= 0;
                        end
```

```
                        else begin
                                image_counter <= image_counter +1;
                        end
                        state <= state_der;
                end else begin
                        state <= state_der_idle;
                        addr_en <= 1'b0;
                end
        end

        state_der_idle: begin
                if (idle_counter < 1) begin
                        addr_en <= 1'b1;
                        state <= state_der_idle;
                        idle_counter <= idle_counter + 1;
                end else begin
                        state <= state_sum_ready;
                end
        end

        state_sum_ready: begin
                for (i=0; i < wsize*wsize; i=i+1) begin
                        kernel0[i] <= kernel_Gxy[i];
                        kernel1[i] <= kernel_Gxy[i];
                        kernel2[i] <= kernel_Gxy[i]; // only used for Sxy
convolution
                end
                image_counter <= 4'b0;
                isFirst <= 1'b1;
                state <= state_sum;
        end
        state_sum: begin
                if (tensor_counter < 10'd900) begin
                        we_R <= 1'b0; // write to M10K for every 8 cycles
                        vga_sram_write <= 1'b0;
                        sram_write <= 1'b0;
                        image_temp0[image_counter] <= q_Ix2;
                        image_temp1[image_counter] <= q_Iy2;
                        image_temp2[image_counter] <= q_Ixy;
                        //image_counter <= image_counter +1;
                        if (image_counter == 8) begin
                                if (isFirst)begin
                                        write_address <= 0;
                                        isFirst <=1'b0;
```

```
                                        end else begin
                                                write_address <= write_address + 1;
                                        end
                                        we_R <= 1'b1;

                                        sram_en <= 1;
                                        /*
                                        sram_write <= 1'b1;
                                        sram_address <= tensor_counter + 1;
                                        if (  $signed(result_R) >
$signed(27'b000000000000011001100110011) )begin
                                                vga_sram_write <= 1'b1;
                                                vga_sram_writedata <=
8'b11111111;
                                                vga_sram_address <=
vga_out_base_address + {22'b0, vga_x_cood} + ({22'b0,vga_y_cood}*640);
                                                // wirte R to hps; otherwise, write 0
                                                sram_writedata <= {5'b0, result_R};
                                        end else begin
                                                sram_writedata <= 32'b0;
                                        end*/

                                        if (vga_x_cood < 30) begin
                                                vga_x_cood <= vga_x_cood + 1;
                                        end else begin
                                                vga_x_cood <= 0;
                                                vga_y_cood <= vga_y_cood + 1;
                                        end
                                        image_counter <= 0;
                                        tensor_counter <= tensor_counter + 1;
                                end else begin
                                        image_counter <= image_counter +1;
                                end
                                if (sram_en) begin
                                        sram_write <= 1'b1;
                                        sram_address <= tensor_counter;
                                        sram_writedata <= {5'b0, result_R};
                                        sram_en <= 1'b0;
                                end
                                state <= state_sum;
                        end else begin
                                we_R <= 1'b0;
                                // let the hps know fpga done
                                sram_address <= 8'd0;
```

```
                                        sram_writedata <= 32'b0;
                                        sram_write <= 1'b1;
                                        state <= state_idle;
                                end
                        end
                        state_idle:begin
                                sram_write <= 1'b0;
                                state <= state_idle;
                        end
                endcase
        end
    end
//endmodule

//=======================================================
//  Structural coding
//=======================================================
// From Qsys

Computer_System The_System (
    ////////////////////////////////////
    // FPGA Side
    ////////////////////////////////////

    // Global signals
    .system_pll_ref_clk_clk                                     (CLOCK_50),
    .system_pll_ref_reset_reset                 (1'b0),

    // SRAM shared block with HPS
    .onchip_sram_s1_address             (sram_address),
    .onchip_sram_s1_clken               (sram_clken),
    .onchip_sram_s1_chipselect          (sram_chipselect),
    .onchip_sram_s1_write               (sram_write),
    .onchip_sram_s1_readdata            (sram_readdata),
    .onchip_sram_s1_writedata           (sram_writedata),
    .onchip_sram_s1_byteenable          (4'b1111),

    //  sram to video
    .onchip_vga_buffer_s1_address    (vga_sram_address),
    .onchip_vga_buffer_s1_clken      (vga_sram_clken),
    .onchip_vga_buffer_s1_chipselect (vga_sram_chipselect),
    .onchip_vga_buffer_s1_write      (vga_sram_write),
    .onchip_vga_buffer_s1_readdata   (),   // never read from vga here
    .onchip_vga_buffer_s1_writedata  (vga_sram_writedata),
```

```
        // AV Config
        .av_config_SCLK                                 (FPGA_I2C_SCLK),
        .av_config_SDAT                                 (FPGA_I2C_SDAT),

        // 50 MHz clock bridge
        .clock_bridge_0_in_clk_clk          (CLOCK_50), //(CLOCK_50),

        // VGA Subsystem
        .vga_pll_ref_clk_clk                        (CLOCK2_50),
        .vga_pll_ref_reset_reset                    (1'b0),
        .vga_CLK
(VGA_CLK),
        .vga_BLANK
(VGA_BLANK_N),
        .vga_SYNC
(VGA_SYNC_N),
        .vga_HS
(VGA_HS),
        .vga_VS
(VGA_VS),
        .vga_R                                          (VGA_R),
        .vga_G                                          (VGA_G),
        .vga_B                                          (VGA_B),

        // SDRAM
        .sdram_clk_clk                                  (DRAM_CLK),
  .sdram_addr                                   (DRAM_ADDR),
        .sdram_ba                                       (DRAM_BA),
        .sdram_cas_n                                (DRAM_CAS_N),
        .sdram_cke                                      (DRAM_CKE),
        .sdram_cs_n
(DRAM_CS_N),
        .sdram_dq                                       (DRAM_DQ),
        .sdram_dqm
({DRAM_UDQM,DRAM_LDQM}),
        .sdram_ras_n                                (DRAM_RAS_N),
        .sdram_we_n
(DRAM_WE_N),

        ////////////////////////////////////
        // HPS Side
        ////////////////////////////////////
        .pio_rst_external_connection_export (pio_rst),
```

```
.pio_zoom_external_connection_export (pio_ci1_new),
.pio_zout_external_connection_export (pio_ci2_new),
.pio_ci3_new_external_connection_export (pio_ci3_new),

// DDR3 SDRAM
.memory_mem_a                   (HPS_DDR3_ADDR),
.memory_mem_ba                  (HPS_DDR3_BA),
.memory_mem_ck                  (HPS_DDR3_CK_P),
.memory_mem_ck_n        (HPS_DDR3_CK_N),
.memory_mem_cke         (HPS_DDR3_CKE),
.memory_mem_cs_n        (HPS_DDR3_CS_N),
.memory_mem_ras_n       (HPS_DDR3_RAS_N),
.memory_mem_cas_n               (HPS_DDR3_CAS_N),
.memory_mem_we_n        (HPS_DDR3_WE_N),
.memory_mem_reset_n     (HPS_DDR3_RESET_N),
.memory_mem_dq                  (HPS_DDR3_DQ),
.memory_mem_dqs         (HPS_DDR3_DQS_P),
.memory_mem_dqs_n               (HPS_DDR3_DQS_N),
.memory_mem_odt         (HPS_DDR3_ODT),
.memory_mem_dm                  (HPS_DDR3_DM),
.memory_oct_rzqin       (HPS_DDR3_RZQ),

// Ethernet
.hps_io_hps_io_gpio_inst_GPIO35   (HPS_ENET_INT_N),
.hps_io_hps_io_emac1_inst_TX_CLK        (HPS_ENET_GTX_CLK),
.hps_io_hps_io_emac1_inst_TXD0   (HPS_ENET_TX_DATA[0]),
.hps_io_hps_io_emac1_inst_TXD1   (HPS_ENET_TX_DATA[1]),
.hps_io_hps_io_emac1_inst_TXD2   (HPS_ENET_TX_DATA[2]),
.hps_io_hps_io_emac1_inst_TXD3   (HPS_ENET_TX_DATA[3]),
.hps_io_hps_io_emac1_inst_RXD0   (HPS_ENET_RX_DATA[0]),
.hps_io_hps_io_emac1_inst_MDIO   (HPS_ENET_MDIO),
.hps_io_hps_io_emac1_inst_MDC           (HPS_ENET_MDC),
.hps_io_hps_io_emac1_inst_RX_CTL        (HPS_ENET_RX_DV),
.hps_io_hps_io_emac1_inst_TX_CTL        (HPS_ENET_TX_EN),
.hps_io_hps_io_emac1_inst_RX_CLK        (HPS_ENET_RX_CLK),
.hps_io_hps_io_emac1_inst_RXD1   (HPS_ENET_RX_DATA[1]),
.hps_io_hps_io_emac1_inst_RXD2   (HPS_ENET_RX_DATA[2]),
.hps_io_hps_io_emac1_inst_RXD3   (HPS_ENET_RX_DATA[3]),

// Flash
.hps_io_hps_io_qspi_inst_IO0        (HPS_FLASH_DATA[0]),
.hps_io_hps_io_qspi_inst_IO1        (HPS_FLASH_DATA[1]),
.hps_io_hps_io_qspi_inst_IO2        (HPS_FLASH_DATA[2]),
.hps_io_hps_io_qspi_inst_IO3        (HPS_FLASH_DATA[3]),
```

```
.hps_io_hps_io_qspi_inst_SS0        (HPS_FLASH_NCSO),
.hps_io_hps_io_qspi_inst_CLK        (HPS_FLASH_DCLK),

// Accelerometer
.hps_io_hps_io_gpio_inst_GPIO61   (HPS_GSENSOR_INT),

//.adc_sclk                 (ADC_SCLK),
//.adc_cs_n                  (ADC_CS_N),
//.adc_dout                  (ADC_DOUT),
//.adc_din                  (ADC_DIN),

// General Purpose I/O
.hps_io_hps_io_gpio_inst_GPIO40   (HPS_GPIO[0]),
.hps_io_hps_io_gpio_inst_GPIO41   (HPS_GPIO[1]),

// I2C
.hps_io_hps_io_gpio_inst_GPIO48   (HPS_I2C_CONTROL),
.hps_io_hps_io_i2c0_inst_SDA            (HPS_I2C1_SDAT),
.hps_io_hps_io_i2c0_inst_SCL            (HPS_I2C1_SCLK),
.hps_io_hps_io_i2c1_inst_SDA            (HPS_I2C2_SDAT),
.hps_io_hps_io_i2c1_inst_SCL            (HPS_I2C2_SCLK),

// Pushbutton
.hps_io_hps_io_gpio_inst_GPIO54   (HPS_KEY),

// LED
.hps_io_hps_io_gpio_inst_GPIO53   (HPS_LED),

// SD Card
.hps_io_hps_io_sdio_inst_CMD       (HPS_SD_CMD),
.hps_io_hps_io_sdio_inst_D0(HPS_SD_DATA[0]),
.hps_io_hps_io_sdio_inst_D1(HPS_SD_DATA[1]),
.hps_io_hps_io_sdio_inst_CLK       (HPS_SD_CLK),
.hps_io_hps_io_sdio_inst_D2(HPS_SD_DATA[2]),
.hps_io_hps_io_sdio_inst_D3(HPS_SD_DATA[3]),

// SPI
.hps_io_hps_io_spim1_inst_CLK           (HPS_SPIM_CLK),
.hps_io_hps_io_spim1_inst_MOSI    (HPS_SPIM_MOSI),
.hps_io_hps_io_spim1_inst_MISO    (HPS_SPIM_MISO),
.hps_io_hps_io_spim1_inst_SS0           (HPS_SPIM_SS),

// UART
.hps_io_hps_io_uart0_inst_RX        (HPS_UART_RX),
```

```
        .hps_io_hps_io_uart0_inst_TX        (HPS_UART_TX),

        // USB
        .hps_io_hps_io_gpio_inst_GPIO09   (HPS_CONV_USB_N),
        .hps_io_hps_io_usb1_inst_D0             (HPS_USB_DATA[0]),
        .hps_io_hps_io_usb1_inst_D1             (HPS_USB_DATA[1]),
        .hps_io_hps_io_usb1_inst_D2             (HPS_USB_DATA[2]),
        .hps_io_hps_io_usb1_inst_D3             (HPS_USB_DATA[3]),
        .hps_io_hps_io_usb1_inst_D4             (HPS_USB_DATA[4]),
        .hps_io_hps_io_usb1_inst_D5             (HPS_USB_DATA[5]),
        .hps_io_hps_io_usb1_inst_D6             (HPS_USB_DATA[6]),
        .hps_io_hps_io_usb1_inst_D7             (HPS_USB_DATA[7]),
        .hps_io_hps_io_usb1_inst_CLK            (HPS_USB_CLKOUT),
        .hps_io_hps_io_usb1_inst_STP            (HPS_USB_STP),
        .hps_io_hps_io_usb1_inst_DIR            (HPS_USB_DIR),
        .hps_io_hps_io_usb1_inst_NXT            (HPS_USB_NXT)
);
endmodule // end top level

//============================================================
// M10K module for testing
//============================================================
// See example 12-16 in
// http://people.ece.cornell.edu/land/courses/ece5760/DE1_SOC/HDL_style_qts_qii51007.pdf
//============================================================

module M10K_256_32(
   output reg [31:0] q,
   input [31:0] d,
   input [7:0] write_address, read_address,
   input we, clk
);
         // force M10K ram style
         // 256 words of 32 bits
   reg [31:0] mem [255:0]  /* synthesis ramstyle = "no_rw_check, M10K" */;

   always @ (posedge clk) begin
      if (we) begin
         mem[write_address] <= d;
                  end
      q <= mem[read_address]; // q doesn't get d in this clock cycle
   end
endmodule
```

```
//=============================================================
// MLAB module for testing
//=============================================================
// See example 12-16 in
// http://people.ece.cornell.edu/land/courses/ece5760/DE1_SOC/HDL_style_qts_qii51007.pdf
//=============================================================
module MLAB_20_32(
	output reg signed [31:0] q,
	input  [31:0] data,
	input [7:0] readaddr, writeaddr,
	input wren, clock
);
	// force MLAB ram style
	// 20 words of 32 bits
	reg signed [31:0] mem [19:0] /* synthesis ramstyle = "no_rw_check, MLAB" */;

	always @ (posedge clock)
	begin
		if (wren) begin
			mem[writeaddr] <= data;
		end
		q <= mem[readaddr];
	end
endmodule

/************************************************************************
 * Following floating point modules written by Bruce Land
 * March 2017
 ************************************************************************/
/************************************************************************
 * Floating Point to 16-bit integer                                     *
 * Combinational
 * Numbers with mag > than +/-32768 get clipped to 32768 or -32768
 ************************************************************************/
 module Int2Fp(
		input signed [15:0]	iInteger,
		output[26:0]	oA
 );
		// output fields
   wire		A_s;
   wire [7:0]  A_e;
   wire [17:0] A_f;

	 wire [15:0] abs_input ;
```

```
        // get output sign bit
        assign A_s = (iInteger < 0);
        // remove sign from input
        assign abs_input = (iInteger < 0)? -iInteger : iInteger ;

        // find the most significant (nonzero) bit
        wire [7:0]  shft_amt;
        assign shft_amt = abs_input[15] ? 8'd3 :
                abs_input[14] ? 8'd4 : abs_input[13] ? 8'd5 :
                abs_input[12] ? 8'd6 : abs_input[11] ? 8'd7 :
                abs_input[10] ? 8'd8 : abs_input[9]  ? 8'd9 :
                abs_input[8]  ? 8'd10 : abs_input[7]  ? 8'd11 :
                abs_input[6]  ? 8'd12 : abs_input[5]  ? 8'd13 :
                abs_input[4]  ? 8'd14 : abs_input[3]  ? 8'd15 :
                abs_input[2]  ? 8'd16 : abs_input[1]  ? 8'd17 :
                abs_input[0]  ? 8'd18 : 8'd19;
        // exponent 127 + (18-shift_amt)
        // 127 is 2^0
        // 18 is amount '1' is shifted
        assign A_e = 127 + 18 - shft_amt ;
        // where the intermediate value is formed
        wire [33:0] shift_buffer ;
        // remember that the high-order '1' is not stored,
        // but is shifted to bit 18
        assign shift_buffer = {16'b0, abs_input} << shft_amt ;
        assign A_f = shift_buffer[17:0];
        assign oA = (iInteger==0)? 27'b0 : {A_s, A_e, A_f};

endmodule //Int2Fp

/***************************************************************************
* Floating Point to 16-bit integer                                         *
* Combinational
* Numbers with mag > than +/-32768 get clipped to 32768 or -32768
***************************************************************************/
module Fp2Int(
                input     [26:0]  iA,
                output reg [15:0]        oInteger
);
                // Extract fields of A and B.
  wire       A_s;
  wire [7:0]  A_e;
  wire [17:0] A_f;
  assign A_s = iA[26];
```

```
  assign A_e = iA[25:18];
  assign A_f = iA[17:0];

        wire [15:0] max_int = 16'h7fff ; //32768
        wire [33:0] shift_buffer ;
        // form (1.A_f) and shift it to postiion
        assign shift_buffer = {15'b0, 1'b1, A_f}<<(A_e-127) ;

        // If exponent less than 127, oInteger=0
        // If exponent greater than 127+14 oInteger=max value
        // Between these two values:
        //      set up input mantissa with 1.mantissa
        //        and the "1." in the lowest bit of an extended word.
        //      shift-left by A_e-127
        // If the sign bit is set, negate oInteger

        always @(*) begin
                if (A_e < 127) oInteger = 16'b0;
                else if (A_e > 141) begin
                        if (A_s) oInteger = -max_int;
                        else     oInteger = max_int;
                end
                else begin
                        if (A_s) oInteger = -shift_buffer[33:18];
                        else     oInteger = shift_buffer[33:18];
                end
        end

endmodule //Fp2Int

/***************************************************************************
 * Floating Point shift                                    *
 * Combinational
 * Negative shift input is right shift
 ***************************************************************************/
 module FpShift(
                input     [26:0]  iA,
                input   [7:0]     iShift,
                output [26:0]   oShifted
 );
                // Extract fields of A and B.
  wire       A_s;
  wire [7:0]  A_e;
  wire [17:0] A_f;
```

```
   assign A_s = iA[26];
   assign A_e = iA[25:18];
   assign A_f = iA[17:0];
          // Flip bit 26
          // zero the output if underflow/overflow
//    assign oShifted = (A_e+iShift<8'd254 && A_e+iShift>8'd2)?
//                                                                    {A_s, A_e+iShift, A_f}
          assign oShifted = {A_s, A_e+iShift, A_f} ;
 endmodule //FpShift

/*************************************************************************
 * Floating Point sign negation                                          *
 * Combinational                                                         *
 *************************************************************************/
 module FpNegate(
                  input     [26:0]  iA,
                  output [26:0]    oNegative
 );
                  // Extract fields of A and B.
   wire       A_s;
   wire [7:0]  A_e;
   wire [17:0] A_f;
   assign A_s = iA[26];
   assign A_e = iA[25:18];
   assign A_f = iA[17:0];
          // Flip bit 26
   assign oNegative = {~A_s, A_e, A_f};
 endmodule //FpNegate

 /*************************************************************************
 * Floating Point absolute                                               *
 * Combinational                                                         *
 *************************************************************************/
 module FpAbs(
                  input     [26:0]  iA,
                  output [26:0]    oAbs
 );
                  // Extract fields of A and B.
   wire       A_s;
   wire [7:0]  A_e;
   wire [17:0] A_f;
   assign A_s = iA[26];
   assign A_e = iA[25:18];
   assign A_f = iA[17:0];
```

```
        // zero bit 26
  assign oAbs = {1'b0, A_e, A_f};
endmodule //Fp absolute

/*****************************************************************************
* Floating Point compare                                        *
* Combinational
* output=1 if A>=B
*****************************************************************************/
module FpCompare(
                input     [26:0]  iA,
                input     [26:0]  iB,
                output reg oA_larger
);
                // Extract fields of A and B.
  wire       A_s;
  wire [7:0]  A_e;
  wire [17:0] A_f;
        wire       B_s;
  wire [7:0]  B_e;
  wire [17:0] B_f;

  assign A_s = iA[26];
  assign A_e = iA[25:18];
  assign A_f = iA[17:0];
        assign B_s = iB[26];
  assign B_e = iB[25:18];
  assign B_f = iB[17:0];

        // Determine which of A, B is larger
        wire A_mag_larger ;
  assign A_mag_larger =(A_e > B_e)                ? 1'b1  :
                ((A_e == B_e) && (A_f >= B_f)) ? 1'b1  :
                1'b0;

        // now do the sign checks
        always @(*) begin
                        if (A_s==0 && B_s==1) begin  // A positive, B negative
                                oA_larger = 1'b1 ;
                        end
                        else if (A_s==1 && B_s==0) begin  // A negative, B positive
                                oA_larger = 1'b0 ;
                        end
                        else if (A_s==0 && B_s==0) begin  // A positive, B positive
```

```
                        oA_larger = A_mag_larger ;
                    end
                    else if (A_s==1 && B_s==1) begin  // A negative, B negative
                        oA_larger = ~A_mag_larger ;
                    end
                    else oA_larger  = 0; // make sure no inferred latch
        end
endmodule //FpCompare

/*************************************************************************
 * Following floating point written by Mark Eiding mje56                 *
 * ECE 5760                                                              *
 * Modified IEEE single precision FP                                     *
 * bit 26:      Sign     (0: pos, 1: neg)                                *
 * bits[25:18]: Exponent (unsigned)                                      *
 * bits[17:0]:  Fraction (unsigned)                                      *
 *  (-1)^SIGN * 2^(EXP-127) * (1+.FRAC)                                  *
 * (http://en.wikipedia.org/wiki/Single-precision_floating-point_format) *
 * Adapted from Skyler Schneider ss868                                   *
 *************************************************************************/
/*************************************************************************
 * Floating Point Fast Inverse Square Root                               *
 * 5-stage pipeline                                                      *
 * http://en.wikipedia.org/wiki/Fast_inverse_square_root                 *
 * Magic number 27'd49920718                                             *
 * 1.5 = 27'd33423360                                                    *
 *************************************************************************/
module FpInvSqrt (
   input            iCLK,
   input     [26:0] iA,
   output    [26:0] oInvSqrt
);

   // Extract fields of A and B.
   wire       A_s;
   wire [7:0]  A_e;
   wire [17:0] A_f;
   assign A_s = iA[26];
   assign A_e = iA[25:18];
   assign A_f = iA[17:0];

   //Stage 1
   wire [26:0] y_1, y_1_out, half_iA_1;
   assign y_1 = 27'd49920718 - (iA>>1);
```

```
    assign half_iA_1 = {A_s, A_e-8'd1,A_f};
    FpMul s1_mult ( .iA(y_1), .iB(y_1), .oProd(y_1_out) );
    //Stage 2
    reg [26:0] y_2, mult_2_in, half_iA_2;
    wire [26:0] y_2_out;
    FpMul s2_mult ( .iA(half_iA_2), .iB(mult_2_in), .oProd(y_2_out) );
    //Stage 3
    reg [26:0] y_3, add_3_in;
    wire [26:0] y_3_out;
    FpAdd s3_add ( .iCLK(iCLK), .iA({~add_3_in[26],add_3_in[25:0]}), .iB(27'd33423360),
.oSum(y_3_out) );
    //Stage 4
    reg [26:0] y_4;
    //Stage 5
    reg [26:0] y_5, mult_5_in;
    FpMul s5_mult ( .iA(y_5), .iB(mult_5_in), .oProd(oInvSqrt) );

    always @(posedge iCLK) begin
    //Stage 1 to 2
    y_2 <= y_1;
    mult_2_in <= y_1_out;
    half_iA_2 <= half_iA_1;
    //Stage 2 to 3
    y_3 <= y_2;
    add_3_in <= y_2_out;
    //Stage 3 to 4
    y_4 <= y_3;
    //Stage 4 to 5
    y_5 <= y_4;
    mult_5_in <= y_3_out;
    end
endmodule

/***************************************************************************
 * Floating Point Multiplier                                               *
 * Combinational                                                           *
 ***************************************************************************/
module FpMul (
    input      [26:0] iA,    // First input
    input      [26:0] iB,    // Second input
    output     [26:0] oProd  // Product
);

    // Extract fields of A and B.
```

```
    wire        A_s;
    wire [7:0]  A_e;
    wire [17:0] A_f;
    wire        B_s;
    wire [7:0]  B_e;
    wire [17:0] B_f;
    assign A_s = iA[26];
    assign A_e = iA[25:18];
    assign A_f = {1'b1, iA[17:1]};
    assign B_s = iB[26];
    assign B_e = iB[25:18];
    assign B_f = {1'b1, iB[17:1]};

    // XOR sign bits to determine product sign.
    wire        oProd_s;
    assign oProd_s = A_s ^ B_s;

    // Multiply the fractions of A and B
    wire [35:0] pre_prod_frac;
    assign pre_prod_frac = A_f * B_f;

    // Add exponents of A and B
    wire [8:0]  pre_prod_exp;
    assign pre_prod_exp = A_e + B_e;

    // If top bit of product frac is 0, shift left one
    wire [7:0]  oProd_e;
    wire [17:0] oProd_f;
    assign oProd_e = pre_prod_frac[35] ? (pre_prod_exp-9'd126) : (pre_prod_exp - 9'd127);
    assign oProd_f = pre_prod_frac[35] ? pre_prod_frac[34:17] : pre_prod_frac[33:16];

    // Detect underflow
    wire        underflow;
    assign underflow = pre_prod_exp < 9'h80;

    // Detect zero conditions (either product frac doesn't start with 1, or underflow)
    assign oProd = underflow        ? 27'b0 :
                   (B_e == 8'd0)    ? 27'b0 :
                   (A_e == 8'd0)    ? 27'b0 :
                   {oProd_s, oProd_e, oProd_f};

endmodule
```

```
/************************************************************************
 * Floating Point Adder                                                 *
 * 2-stage pipeline                                                     *
 ************************************************************************/
module FpAdd (
   input            iCLK,
   input      [26:0] iA,
   input      [26:0] iB,
   output reg [26:0] oSum
);

   // Extract fields of A and B.
   wire        A_s;
   wire [7:0]  A_e;
   wire [17:0] A_f;
   wire        B_s;
   wire [7:0]  B_e;
   wire [17:0] B_f;
   assign A_s = iA[26];
   assign A_e = iA[25:18];
   assign A_f = {1'b1, iA[17:1]};
   assign B_s = iB[26];
   assign B_e = iB[25:18];
   assign B_f = {1'b1, iB[17:1]};
   wire A_larger;

   // Shift fractions of A and B so that they align.
   wire [7:0]  exp_diff_A;
   wire [7:0]  exp_diff_B;
   wire [7:0]  larger_exp;
   wire [36:0] A_f_shifted;
   wire [36:0] B_f_shifted;

   assign exp_diff_A = B_e - A_e; // if B bigger
   assign exp_diff_B = A_e - B_e; // if A bigger

   assign larger_exp = (B_e > A_e) ? B_e : A_e;

   assign A_f_shifted = A_larger            ? {1'b0,  A_f, 18'b0} :
                        (exp_diff_A > 9'd35) ? 37'b0 :
                        ({1'b0, A_f, 18'b0} >> exp_diff_A);
   assign B_f_shifted = ~A_larger           ? {1'b0,  B_f, 18'b0} :
                        (exp_diff_B > 9'd35) ? 37'b0 :
                        ({1'b0, B_f, 18'b0} >> exp_diff_B);
```

```
// Determine which of A, B is larger
assign A_larger =    (A_e > B_e)                 ? 1'b1  :
                     ((A_e == B_e) && (A_f > B_f)) ? 1'b1  :
                     1'b0;

// Calculate sum or difference of shifted fractions.
wire [36:0] pre_sum;
assign pre_sum = ((A_s^B_s) &  A_larger) ? A_f_shifted - B_f_shifted :
                 ((A_s^B_s) & ~A_larger) ? B_f_shifted - A_f_shifted :
                 A_f_shifted + B_f_shifted;

// buffer midway results
reg  [36:0] buf_pre_sum;
reg  [7:0]  buf_larger_exp;
reg         buf_A_e_zero;
reg         buf_B_e_zero;
reg  [26:0] buf_A;
reg  [26:0] buf_B;
reg         buf_oSum_s;
always @(posedge iCLK) begin
   buf_pre_sum    <= pre_sum;
   buf_larger_exp <= larger_exp;
   buf_A_e_zero   <= (A_e == 8'b0);
   buf_B_e_zero   <= (B_e == 8'b0);
   buf_A          <= iA;
   buf_B          <= iB;
   buf_oSum_s     <= A_larger ? A_s : B_s;
end

// Convert to positive fraction and a sign bit.
wire [36:0] pre_frac;
assign pre_frac = buf_pre_sum;

// Determine output fraction and exponent change with position of first 1.
wire [17:0] oSum_f;
wire [7:0]  shft_amt;
assign shft_amt = pre_frac[36] ? 8'd0  : pre_frac[35] ? 8'd1  :
                  pre_frac[34] ? 8'd2  : pre_frac[33] ? 8'd3  :
                  pre_frac[32] ? 8'd4  : pre_frac[31] ? 8'd5  :
                  pre_frac[30] ? 8'd6  : pre_frac[29] ? 8'd7  :
                  pre_frac[28] ? 8'd8  : pre_frac[27] ? 8'd9  :
                  pre_frac[26] ? 8'd10 : pre_frac[25] ? 8'd11 :
                  pre_frac[24] ? 8'd12 : pre_frac[23] ? 8'd13 :
```

```
                pre_frac[22] ? 8'd14 : pre_frac[21] ? 8'd15 :
                pre_frac[20] ? 8'd16 : pre_frac[19] ? 8'd17 :
                pre_frac[18] ? 8'd18 : pre_frac[17] ? 8'd19 :
                pre_frac[16] ? 8'd20 : pre_frac[15] ? 8'd21 :
                pre_frac[14] ? 8'd22 : pre_frac[13] ? 8'd23 :
                pre_frac[12] ? 8'd24 : pre_frac[11] ? 8'd25 :
                pre_frac[10] ? 8'd26 : pre_frac[9]  ? 8'd27 :
                pre_frac[8]  ? 8'd28 : pre_frac[7]  ? 8'd29 :
                pre_frac[6]  ? 8'd30 : pre_frac[5]  ? 8'd31 :
                pre_frac[4]  ? 8'd32 : pre_frac[3]  ? 8'd33 :
                pre_frac[2]  ? 8'd34 : pre_frac[1]  ? 8'd35 :
                pre_frac[0]  ? 8'd36 : 8'd37;

    wire [53:0] pre_frac_shft, uflow_shift;
        // the shift +1 is because high order bit is not stored, but implied
    assign pre_frac_shft = {pre_frac, 17'b0} << (shft_amt+1); //? shft_amt+1
        assign uflow_shift = {pre_frac, 17'b0} << (shft_amt); //? shft_amt for overflow
    assign oSum_f = pre_frac_shft[53:36];

    wire [7:0] oSum_e;
    assign oSum_e = buf_larger_exp - shft_amt + 8'b1;

    // Detect underflow
    wire underflow;
        // this incorrectly sets uflow for 10-10.1
    //assign underflow = ~oSum_e[7] && buf_larger_exp[7] && (shft_amt != 8'b0);

        // if top bit of matissa is not set, then denorm
        assign underflow = ~uflow_shift[53];

        always @(posedge iCLK) begin
                    oSum <= (buf_A_e_zero && buf_B_e_zero)    ? 27'b0 :
            buf_A_e_zero                ? buf_B :
            buf_B_e_zero                ? buf_A :
            underflow                  ? 27'b0 :
            (pre_frac == 0)             ? 27'b0 :
            {buf_oSum_s, oSum_e, oSum_f};
        end //output update
endmodule

/// end ////////////////////////////////////////////////////////////////
module adder(a,b,c,out);
        input  signed [26:0] a,b,c;
        output signed [26:0] out;
```

```
        assign out = a+b+c;
endmodule


module signed_mult(a,b,out);
        input  signed [26:0] a,b;
        output signed [26:0] out;

        wire   signed [53:0] mult_out;
        assign mult_out = a * b;
        //4.23
   //assign out      ={mult_out[53],mult_out[48:23]};
   //6.21
   assign out = {mult_out[53],mult_out[47:21]};
endmodule

module response(R, x2, y2, xy, rst);
        output signed [26:0] R;
        input  signed [26:0] x2, y2, xy;
        input rst;

        wire signed [26:0] temp_det0, temp_det1, temp_trace0, temp_trace1;
        wire signed [26:0] k;

        assign k = 27'b000000000010100011110101110; // This number is around 0.05; should
be in the range of [0.04, 0.06]

        signed_mult det0(.a(x2>>>2), .b(y2>>>2), .out(temp_det0));
        signed_mult det1(.a(xy>>>2), .b(xy>>>2), .out(temp_det1));
        signed_mult trace0(.a((x2+y2)>>>2), .b((x2+y2)>>>2), .out(temp_trace0));
        signed_mult trace1(.a(k), .b(temp_trace0), .out(temp_trace1));
        assign R = temp_det0 - temp_det1 - temp_trace1;
endmodule

//This style will be synthesize to M10K block
module dual_clock_ram(q, d, write_address, read_address, we, clk1, clk2);
        parameter  isize=30;
        output reg signed [26:0] q;
        input      signed [26:0] d;
        input      [10:0]  write_address, read_address; // for 900 nodes in one column
        input              we, clk1, clk2;

        reg   [10:0] read_address_reg;
        reg   signed [27:0] mem [0:(isize+2)*(isize+2)-1];
```

```
	always @ (posedge clk1)
	begin
		if (we)
		mem[write_address] <= d;
	end
	always @ (posedge clk2) begin
		q <= mem[read_address_reg];
		read_address_reg <= read_address;
	end
endmodule
/*
module sync_rom_image(clk,address,mem_out);
	input		clk;
	input	[10:0] address;
	output	[26:0] mem_out;
	reg signed [26:0] mem_out;
	always@(posedge clk) begin
		case(address)

		11'h0: mem_out = 27'b000000000000000000000000000;
		11'h1: mem_out = 27'b000000000000000000000000000;
		11'h2: mem_out = 27'b000000000000000000000000000;
		11'h3: mem_out = 27'b000000000000000000000000000;
		11'h4: mem_out = 27'b000000000000000000000000000;
		11'h5: mem_out = 27'b000000000000000000000000000;
		11'h6: mem_out = 27'b000000000000000000000000000;
		11'h7: mem_out = 27'b000000000000000000000000000;
		11'h8: mem_out = 27'b000000000000000000000000000;
		11'h9: mem_out = 27'b000000000000000000000000000;
		11'hA: mem_out = 27'b000000000000000000000000000;
		11'hB: mem_out = 27'b000000000000000000000000000;
		11'hC: mem_out = 27'b000000000000000000000000000;
		11'hD: mem_out = 27'b000000000000000000000000000;
		11'hE: mem_out = 27'b000000000000000000000000000;
		11'hF: mem_out = 27'b000000000000000000000000000;
		11'h10: mem_out = 27'b000000000000000000000000000;
		11'h11: mem_out = 27'b000000000000000000000000000;
		11'h12: mem_out = 27'b000000000000000000000000000;
		11'h13: mem_out = 27'b000000000000000000000000000;
		11'h14: mem_out = 27'b000000000000000000000000000;
		11'h15: mem_out = 27'b000000000000000000000000000;
		11'h16: mem_out = 27'b000000000000000000000000000;
		11'h17: mem_out = 27'b000000000000000000000000000;
		11'h18: mem_out = 27'b000000000000000000000000000;
```

```
11'h19: mem_out = 27'b000000000000000000000000000;
11'h1A: mem_out = 27'b000000000000000000000000000;
11'h1B: mem_out = 27'b000000000000000000000000000;
11'h1C: mem_out = 27'b000000000000000000000000000;
11'h1D: mem_out = 27'b000000000000000000000000000;
11'h1E: mem_out = 27'b000000000000000000000000000;
11'h1F: mem_out = 27'b000000000000000000000000000;
11'h20: mem_out = 27'b000000000000000000000000000;
11'h21: mem_out = 27'b000000000000000000000000000;
11'h22: mem_out = 27'b000000000000000000000000000;
11'h23: mem_out = 27'b000000000000000000000000000;
11'h24: mem_out = 27'b000000000000000000000000000;
11'h25: mem_out = 27'b000000000000000000000000000;
11'h26: mem_out = 27'b000000000000000000000000000;
11'h27: mem_out = 27'b000000000000000000000000000;
11'h28: mem_out = 27'b000000000000000000000000000;
11'h29: mem_out = 27'b000000000000000000000000000;
11'h2A: mem_out = 27'b000000000000000000000000000;
11'h2B: mem_out = 27'b000000000000000000000000000;
11'h2C: mem_out = 27'b000000000000000000000000000;
11'h2D: mem_out = 27'b000000000000000000000000000;
11'h2E: mem_out = 27'b000000000000000000000000000;
11'h2F: mem_out = 27'b000000000000000000000000000;
11'h30: mem_out = 27'b000000000000000000000000000;
11'h31: mem_out = 27'b000000000000000000000000000;
11'h32: mem_out = 27'b000000000000000000000000000;
11'h33: mem_out = 27'b000000000000000000000000000;
11'h34: mem_out = 27'b000000000000000000000000000;
11'h35: mem_out = 27'b000000000000000000000000000;
11'h36: mem_out = 27'b000000000000000000000000000;
11'h37: mem_out = 27'b000000000000000000000000000;
11'h38: mem_out = 27'b000000000000000000000000000;
11'h39: mem_out = 27'b000000000000000000000000000;
11'h3A: mem_out = 27'b000000000000000000000000000;
11'h3B: mem_out = 27'b000000000000000000000000000;
11'h3C: mem_out = 27'b000000000000000000000000000;
11'h3D: mem_out = 27'b000000000000000000000000000;
11'h3E: mem_out = 27'b000000000000000000000000000;
11'h3F: mem_out = 27'b000000000000000000000000000;
11'h40: mem_out = 27'b000000000000000000000000000;
11'h41: mem_out = 27'b000000000000000000000000000;
11'h42: mem_out = 27'b000001000000000000000000000;
11'h43: mem_out = 27'b000001000000000000000000000;
11'h44: mem_out = 27'b000001000000000000000000000;
```

```
11'h45: mem_out = 27'b000001000000000000000000000;
11'h46: mem_out = 27'b000001000000000000000000000;
11'h47: mem_out = 27'b000001000000000000000000000;
11'h48: mem_out = 27'b000001000000000000000000000;
11'h49: mem_out = 27'b000001000000000000000000000;
11'h4A: mem_out = 27'b000001000000000000000000000;
11'h4B: mem_out = 27'b000001000000000000000000000;
11'h4C: mem_out = 27'b000000000000000000000000000;
11'h4D: mem_out = 27'b000000000000000000000000000;
11'h4E: mem_out = 27'b000000000000000000000000000;
11'h4F: mem_out = 27'b000000000000000000000000000;
11'h50: mem_out = 27'b000000000000000000000000000;
11'h51: mem_out = 27'b000000000000000000000000000;
11'h52: mem_out = 27'b000000000000000000000000000;
11'h53: mem_out = 27'b000000000000000000000000000;
11'h54: mem_out = 27'b000001000000000000000000000;
11'h55: mem_out = 27'b000001000000000000000000000;
11'h56: mem_out = 27'b000001000000000000000000000;
11'h57: mem_out = 27'b000001000000000000000000000;
11'h58: mem_out = 27'b000001000000000000000000000;
11'h59: mem_out = 27'b000001000000000000000000000;
11'h5A: mem_out = 27'b000001000000000000000000000;
11'h5B: mem_out = 27'b000001000000000000000000000;
11'h5C: mem_out = 27'b000001000000000000000000000;
11'h5D: mem_out = 27'b000001000000000000000000000;
11'h5E: mem_out = 27'b000000000000000000000000000;
11'h5F: mem_out = 27'b000000000000000000000000000;
11'h60: mem_out = 27'b000000000000000000000000000;
11'h61: mem_out = 27'b000000000000000000000000000;
11'h62: mem_out = 27'b000001000000000000000000000;
11'h63: mem_out = 27'b000001000000000000000000000;
11'h64: mem_out = 27'b000001000000000000000000000;
11'h65: mem_out = 27'b000001000000000000000000000;
11'h66: mem_out = 27'b000001000000000000000000000;
11'h67: mem_out = 27'b000001000000000000000000000;
11'h68: mem_out = 27'b000001000000000000000000000;
11'h69: mem_out = 27'b000001000000000000000000000;
11'h6A: mem_out = 27'b000001000000000000000000000;
11'h6B: mem_out = 27'b000001000000000000000000000;
11'h6C: mem_out = 27'b000000000000000000000000000;
11'h6D: mem_out = 27'b000000000000000000000000000;
11'h6E: mem_out = 27'b000000000000000000000000000;
11'h6F: mem_out = 27'b000000000000000000000000000;
11'h70: mem_out = 27'b000000000000000000000000000;
```

```
11'h71: mem_out = 27'b000000000000000000000000000;
11'h72: mem_out = 27'b000000000000000000000000000;
11'h73: mem_out = 27'b000000000000000000000000000;
11'h74: mem_out = 27'b000001000000000000000000000;
11'h75: mem_out = 27'b000001000000000000000000000;
11'h76: mem_out = 27'b000001000000000000000000000;
11'h77: mem_out = 27'b000001000000000000000000000;
11'h78: mem_out = 27'b000001000000000000000000000;
11'h79: mem_out = 27'b000001000000000000000000000;
11'h7A: mem_out = 27'b000001000000000000000000000;
11'h7B: mem_out = 27'b000001000000000000000000000;
11'h7C: mem_out = 27'b000001000000000000000000000;
11'h7D: mem_out = 27'b000001000000000000000000000;
11'h7E: mem_out = 27'b000000000000000000000000000;
11'h7F: mem_out = 27'b000000000000000000000000000;
11'h80: mem_out = 27'b000000000000000000000000000;
11'h81: mem_out = 27'b000000000000000000000000000;
11'h82: mem_out = 27'b000000000000000000000000000;
11'h83: mem_out = 27'b000000000000000000000000000;
11'h84: mem_out = 27'b000000000000000000000000000;
11'h85: mem_out = 27'b000001000000000000000000000;
11'h86: mem_out = 27'b000001000000000000000000000;
11'h87: mem_out = 27'b000001000000000000000000000;
11'h88: mem_out = 27'b000001000000000000000000000;
11'h89: mem_out = 27'b000000000000000000000000000;
11'h8A: mem_out = 27'b000000000000000000000000000;
11'h8B: mem_out = 27'b000000000000000000000000000;
11'h8C: mem_out = 27'b000000000000000000000000000;
11'h8D: mem_out = 27'b000000000000000000000000000;
11'h8E: mem_out = 27'b000000000000000000000000000;
11'h8F: mem_out = 27'b000000000000000000000000000;
11'h90: mem_out = 27'b000000000000000000000000000;
11'h91: mem_out = 27'b000000000000000000000000000;
11'h92: mem_out = 27'b000000000000000000000000000;
11'h93: mem_out = 27'b000000000000000000000000000;
11'h94: mem_out = 27'b000000000000000000000000000;
11'h95: mem_out = 27'b000000000000000000000000000;
11'h96: mem_out = 27'b000000000000000000000000000;
11'h97: mem_out = 27'b000001000000000000000000000;
11'h98: mem_out = 27'b000001000000000000000000000;
11'h99: mem_out = 27'b000001000000000000000000000;
11'h9A: mem_out = 27'b000001000000000000000000000;
11'h9B: mem_out = 27'b000000000000000000000000000;
11'h9C: mem_out = 27'b000000000000000000000000000;
```

```
11'h9D: mem_out = 27'b000000000000000000000000000;
11'h9E: mem_out = 27'b000000000000000000000000000;
11'h9F: mem_out = 27'b000000000000000000000000000;
11'hA0: mem_out = 27'b000000000000000000000000000;
11'hA1: mem_out = 27'b000000000000000000000000000;
11'hA2: mem_out = 27'b000000000000000000000000000;
11'hA3: mem_out = 27'b000000000000000000000000000;
11'hA4: mem_out = 27'b000000000000000000000000000;
11'hA5: mem_out = 27'b000001000000000000000000000;
11'hA6: mem_out = 27'b000001000000000000000000000;
11'hA7: mem_out = 27'b000001000000000000000000000;
11'hA8: mem_out = 27'b000001000000000000000000000;
11'hA9: mem_out = 27'b000000000000000000000000000;
11'hAA: mem_out = 27'b000000000000000000000000000;
11'hAB: mem_out = 27'b000000000000000000000000000;
11'hAC: mem_out = 27'b000000000000000000000000000;
11'hAD: mem_out = 27'b000000000000000000000000000;
11'hAE: mem_out = 27'b000000000000000000000000000;
11'hAF: mem_out = 27'b000000000000000000000000000;
11'hB0: mem_out = 27'b000000000000000000000000000;
11'hB1: mem_out = 27'b000000000000000000000000000;
11'hB2: mem_out = 27'b000000000000000000000000000;
11'hB3: mem_out = 27'b000000000000000000000000000;
11'hB4: mem_out = 27'b000000000000000000000000000;
11'hB5: mem_out = 27'b000000000000000000000000000;
11'hB6: mem_out = 27'b000000000000000000000000000;
11'hB7: mem_out = 27'b000001000000000000000000000;
11'hB8: mem_out = 27'b000001000000000000000000000;
11'hB9: mem_out = 27'b000001000000000000000000000;
11'hBA: mem_out = 27'b000001000000000000000000000;
11'hBB: mem_out = 27'b000000000000000000000000000;
11'hBC: mem_out = 27'b000000000000000000000000000;
11'hBD: mem_out = 27'b000000000000000000000000000;
11'hBE: mem_out = 27'b000000000000000000000000000;
11'hBF: mem_out = 27'b000000000000000000000000000;
11'hC0: mem_out = 27'b000000000000000000000000000;
11'hC1: mem_out = 27'b000000000000000000000000000;
11'hC2: mem_out = 27'b000000000000000000000000000;
11'hC3: mem_out = 27'b000000000000000000000000000;
11'hC4: mem_out = 27'b000000000000000000000000000;
11'hC5: mem_out = 27'b000001000000000000000000000;
11'hC6: mem_out = 27'b000001000000000000000000000;
11'hC7: mem_out = 27'b000001000000000000000000000;
11'hC8: mem_out = 27'b000001000000000000000000000;
```

```
11'hC9: mem_out = 27'b000000000000000000000000000;
11'hCA: mem_out = 27'b000000000000000000000000000;
11'hCB: mem_out = 27'b000000000000000000000000000;
11'hCC: mem_out = 27'b000000000000000000000000000;
11'hCD: mem_out = 27'b000000000000000000000000000;
11'hCE: mem_out = 27'b000000000000000000000000000;
11'hCF: mem_out = 27'b000000000000000000000000000;
11'hD0: mem_out = 27'b000000000000000000000000000;
11'hD1: mem_out = 27'b000000000000000000000000000;
11'hD2: mem_out = 27'b000000000000000000000000000;
11'hD3: mem_out = 27'b000000000000000000000000000;
11'hD4: mem_out = 27'b000000000000000000000000000;
11'hD5: mem_out = 27'b000000000000000000000000000;
11'hD6: mem_out = 27'b000000000000000000000000000;
11'hD7: mem_out = 27'b000001000000000000000000000;
11'hD8: mem_out = 27'b000001000000000000000000000;
11'hD9: mem_out = 27'b000001000000000000000000000;
11'hDA: mem_out = 27'b000001000000000000000000000;
11'hDB: mem_out = 27'b000000000000000000000000000;
11'hDC: mem_out = 27'b000000000000000000000000000;
11'hDD: mem_out = 27'b000000000000000000000000000;
11'hDE: mem_out = 27'b000000000000000000000000000;
11'hDF: mem_out = 27'b000000000000000000000000000;
11'hE0: mem_out = 27'b000000000000000000000000000;
11'hE1: mem_out = 27'b000000000000000000000000000;
11'hE2: mem_out = 27'b000000000000000000000000000;
11'hE3: mem_out = 27'b000000000000000000000000000;
11'hE4: mem_out = 27'b000000000000000000000000000;
11'hE5: mem_out = 27'b000001000000000000000000000;
11'hE6: mem_out = 27'b000001000000000000000000000;
11'hE7: mem_out = 27'b000001000000000000000000000;
11'hE8: mem_out = 27'b000001000000000000000000000;
11'hE9: mem_out = 27'b000000000000000000000000000;
11'hEA: mem_out = 27'b000000000000000000000000000;
11'hEB: mem_out = 27'b000000000000000000000000000;
11'hEC: mem_out = 27'b000000000000000000000000000;
11'hED: mem_out = 27'b000000000000000000000000000;
11'hEE: mem_out = 27'b000000000000000000000000000;
11'hEF: mem_out = 27'b000000000000000000000000000;
11'hF0: mem_out = 27'b000000000000000000000000000;
11'hF1: mem_out = 27'b000000000000000000000000000;
11'hF2: mem_out = 27'b000000000000000000000000000;
11'hF3: mem_out = 27'b000000000000000000000000000;
11'hF4: mem_out = 27'b000000000000000000000000000;
```

```
11'hF5: mem_out = 27'b000000000000000000000000000;
11'hF6: mem_out = 27'b000000000000000000000000000;
11'hF7: mem_out = 27'b000001000000000000000000000;
11'hF8: mem_out = 27'b000001000000000000000000000;
11'hF9: mem_out = 27'b000001000000000000000000000;
11'hFA: mem_out = 27'b000001000000000000000000000;
11'hFB: mem_out = 27'b000000000000000000000000000;
11'hFC: mem_out = 27'b000000000000000000000000000;
11'hFD: mem_out = 27'b000000000000000000000000000;
11'hFE: mem_out = 27'b000000000000000000000000000;
11'hFF: mem_out = 27'b000000000000000000000000000;
11'h100: mem_out = 27'b000000000000000000000000000;
11'h101: mem_out = 27'b000000000000000000000000000;
11'h102: mem_out = 27'b000000000000000000000000000;
11'h103: mem_out = 27'b000000000000000000000000000;
11'h104: mem_out = 27'b000000000000000000000000000;
11'h105: mem_out = 27'b000001000000000000000000000;
11'h106: mem_out = 27'b000001000000000000000000000;
11'h107: mem_out = 27'b000001000000000000000000000;
11'h108: mem_out = 27'b000001000000000000000000000;
11'h109: mem_out = 27'b000000000000000000000000000;
11'h10A: mem_out = 27'b000000000000000000000000000;
11'h10B: mem_out = 27'b000000000000000000000000000;
11'h10C: mem_out = 27'b000000000000000000000000000;
11'h10D: mem_out = 27'b000000000000000000000000000;
11'h10E: mem_out = 27'b000000000000000000000000000;
11'h10F: mem_out = 27'b000000000000000000000000000;
11'h110: mem_out = 27'b000000000000000000000000000;
11'h111: mem_out = 27'b000000000000000000000000000;
11'h112: mem_out = 27'b000000000000000000000000000;
11'h113: mem_out = 27'b000000000000000000000000000;
11'h114: mem_out = 27'b000000000000000000000000000;
11'h115: mem_out = 27'b000000000000000000000000000;
11'h116: mem_out = 27'b000000000000000000000000000;
11'h117: mem_out = 27'b000001000000000000000000000;
11'h118: mem_out = 27'b000001000000000000000000000;
11'h119: mem_out = 27'b000001000000000000000000000;
11'h11A: mem_out = 27'b000001000000000000000000000;
11'h11B: mem_out = 27'b000000000000000000000000000;
11'h11C: mem_out = 27'b000000000000000000000000000;
11'h11D: mem_out = 27'b000000000000000000000000000;
11'h11E: mem_out = 27'b000000000000000000000000000;
11'h11F: mem_out = 27'b000000000000000000000000000;
11'h120: mem_out = 27'b000000000000000000000000000;
```

```
11'h121: mem_out = 27'b000000000000000000000000000;
11'h122: mem_out = 27'b000000000000000000000000000;
11'h123: mem_out = 27'b000000000000000000000000000;
11'h124: mem_out = 27'b000000000000000000000000000;
11'h125: mem_out = 27'b000001000000000000000000000;
11'h126: mem_out = 27'b000001000000000000000000000;
11'h127: mem_out = 27'b000001000000000000000000000;
11'h128: mem_out = 27'b000001000000000000000000000;
11'h129: mem_out = 27'b000000000000000000000000000;
11'h12A: mem_out = 27'b000000000000000000000000000;
11'h12B: mem_out = 27'b000000000000000000000000000;
11'h12C: mem_out = 27'b000000000000000000000000000;
11'h12D: mem_out = 27'b000000000000000000000000000;
11'h12E: mem_out = 27'b000000000000000000000000000;
11'h12F: mem_out = 27'b000000000000000000000000000;
11'h130: mem_out = 27'b000000000000000000000000000;
11'h131: mem_out = 27'b000000000000000000000000000;
11'h132: mem_out = 27'b000000000000000000000000000;
11'h133: mem_out = 27'b000000000000000000000000000;
11'h134: mem_out = 27'b000000000000000000000000000;
11'h135: mem_out = 27'b000000000000000000000000000;
11'h136: mem_out = 27'b000000000000000000000000000;
11'h137: mem_out = 27'b000001000000000000000000000;
11'h138: mem_out = 27'b000001000000000000000000000;
11'h139: mem_out = 27'b000001000000000000000000000;
11'h13A: mem_out = 27'b000001000000000000000000000;
11'h13B: mem_out = 27'b000000000000000000000000000;
11'h13C: mem_out = 27'b000000000000000000000000000;
11'h13D: mem_out = 27'b000000000000000000000000000;
11'h13E: mem_out = 27'b000000000000000000000000000;
11'h13F: mem_out = 27'b000000000000000000000000000;
11'h140: mem_out = 27'b000000000000000000000000000;
11'h141: mem_out = 27'b000000000000000000000000000;
11'h142: mem_out = 27'b000000000000000000000000000;
11'h143: mem_out = 27'b000000000000000000000000000;
11'h144: mem_out = 27'b000000000000000000000000000;
11'h145: mem_out = 27'b000001000000000000000000000;
11'h146: mem_out = 27'b000001000000000000000000000;
11'h147: mem_out = 27'b000001000000000000000000000;
11'h148: mem_out = 27'b000001000000000000000000000;
11'h149: mem_out = 27'b000000000000000000000000000;
11'h14A: mem_out = 27'b000000000000000000000000000;
11'h14B: mem_out = 27'b000000000000000000000000000;
11'h14C: mem_out = 27'b000000000000000000000000000;
```

```
11'h14D: mem_out = 27'b000000000000000000000000000;
11'h14E: mem_out = 27'b000000000000000000000000000;
11'h14F: mem_out = 27'b000000000000000000000000000;
11'h150: mem_out = 27'b000000000000000000000000000;
11'h151: mem_out = 27'b000000000000000000000000000;
11'h152: mem_out = 27'b000000000000000000000000000;
11'h153: mem_out = 27'b000000000000000000000000000;
11'h154: mem_out = 27'b000000000000000000000000000;
11'h155: mem_out = 27'b000000000000000000000000000;
11'h156: mem_out = 27'b000000000000000000000000000;
11'h157: mem_out = 27'b000001000000000000000000000;
11'h158: mem_out = 27'b000001000000000000000000000;
11'h159: mem_out = 27'b000001000000000000000000000;
11'h15A: mem_out = 27'b000001000000000000000000000;
11'h15B: mem_out = 27'b000000000000000000000000000;
11'h15C: mem_out = 27'b000000000000000000000000000;
11'h15D: mem_out = 27'b000000000000000000000000000;
11'h15E: mem_out = 27'b000000000000000000000000000;
11'h15F: mem_out = 27'b000000000000000000000000000;
11'h160: mem_out = 27'b000000000000000000000000000;
11'h161: mem_out = 27'b000000000000000000000000000;
11'h162: mem_out = 27'b000000000000000000000000000;
11'h163: mem_out = 27'b000000000000000000000000000;
11'h164: mem_out = 27'b000000000000000000000000000;
11'h165: mem_out = 27'b000001000000000000000000000;
11'h166: mem_out = 27'b000001000000000000000000000;
11'h167: mem_out = 27'b000001000000000000000000000;
11'h168: mem_out = 27'b000001000000000000000000000;
11'h169: mem_out = 27'b000000000000000000000000000;
11'h16A: mem_out = 27'b000000000000000000000000000;
11'h16B: mem_out = 27'b000000000000000000000000000;
11'h16C: mem_out = 27'b000000000000000000000000000;
11'h16D: mem_out = 27'b000000000000000000000000000;
11'h16E: mem_out = 27'b000000000000000000000000000;
11'h16F: mem_out = 27'b000000000000000000000000000;
11'h170: mem_out = 27'b000000000000000000000000000;
11'h171: mem_out = 27'b000000000000000000000000000;
11'h172: mem_out = 27'b000000000000000000000000000;
11'h173: mem_out = 27'b000000000000000000000000000;
11'h174: mem_out = 27'b000000000000000000000000000;
11'h175: mem_out = 27'b000000000000000000000000000;
11'h176: mem_out = 27'b000000000000000000000000000;
11'h177: mem_out = 27'b000001000000000000000000000;
11'h178: mem_out = 27'b000001000000000000000000000;
```

```
11'h179: mem_out = 27'b000001000000000000000000000;
11'h17A: mem_out = 27'b000001000000000000000000000;
11'h17B: mem_out = 27'b000000000000000000000000000;
11'h17C: mem_out = 27'b000000000000000000000000000;
11'h17D: mem_out = 27'b000000000000000000000000000;
11'h17E: mem_out = 27'b000000000000000000000000000;
11'h17F: mem_out = 27'b000000000000000000000000000;
11'h180: mem_out = 27'b000000000000000000000000000;
11'h181: mem_out = 27'b000000000000000000000000000;
11'h182: mem_out = 27'b000000000000000000000000000;
11'h183: mem_out = 27'b000000000000000000000000000;
11'h184: mem_out = 27'b000000000000000000000000000;
11'h185: mem_out = 27'b000001000000000000000000000;
11'h186: mem_out = 27'b000001000000000000000000000;
11'h187: mem_out = 27'b000001000000000000000000000;
11'h188: mem_out = 27'b000001000000000000000000000;
11'h189: mem_out = 27'b000000000000000000000000000;
11'h18A: mem_out = 27'b000000000000000000000000000;
11'h18B: mem_out = 27'b000000000000000000000000000;
11'h18C: mem_out = 27'b000000000000000000000000000;
11'h18D: mem_out = 27'b000000000000000000000000000;
11'h18E: mem_out = 27'b000000000000000000000000000;
11'h18F: mem_out = 27'b000000000000000000000000000;
11'h190: mem_out = 27'b000000000000000000000000000;
11'h191: mem_out = 27'b000000000000000000000000000;
11'h192: mem_out = 27'b000000000000000000000000000;
11'h193: mem_out = 27'b000000000000000000000000000;
11'h194: mem_out = 27'b000000000000000000000000000;
11'h195: mem_out = 27'b000000000000000000000000000;
11'h196: mem_out = 27'b000000000000000000000000000;
11'h197: mem_out = 27'b000001000000000000000000000;
11'h198: mem_out = 27'b000001000000000000000000000;
11'h199: mem_out = 27'b000001000000000000000000000;
11'h19A: mem_out = 27'b000001000000000000000000000;
11'h19B: mem_out = 27'b000000000000000000000000000;
11'h19C: mem_out = 27'b000000000000000000000000000;
11'h19D: mem_out = 27'b000000000000000000000000000;
11'h19E: mem_out = 27'b000000000000000000000000000;
11'h19F: mem_out = 27'b000000000000000000000000000;
11'h1A0: mem_out = 27'b000000000000000000000000000;
11'h1A1: mem_out = 27'b000000000000000000000000000;
11'h1A2: mem_out = 27'b000000000000000000000000000;
11'h1A3: mem_out = 27'b000000000000000000000000000;
11'h1A4: mem_out = 27'b000000000000000000000000000;
```

```
11'h1A5: mem_out = 27'b000001000000000000000000000;
11'h1A6: mem_out = 27'b000001000000000000000000000;
11'h1A7: mem_out = 27'b000001000000000000000000000;
11'h1A8: mem_out = 27'b000001000000000000000000000;
11'h1A9: mem_out = 27'b000000000000000000000000000;
11'h1AA: mem_out = 27'b000000000000000000000000000;
11'h1AB: mem_out = 27'b000000000000000000000000000;
11'h1AC: mem_out = 27'b000000000000000000000000000;
11'h1AD: mem_out = 27'b000000000000000000000000000;
11'h1AE: mem_out = 27'b000000000000000000000000000;
11'h1AF: mem_out = 27'b000000000000000000000000000;
11'h1B0: mem_out = 27'b000000000000000000000000000;
11'h1B1: mem_out = 27'b000000000000000000000000000;
11'h1B2: mem_out = 27'b000000000000000000000000000;
11'h1B3: mem_out = 27'b000000000000000000000000000;
11'h1B4: mem_out = 27'b000000000000000000000000000;
11'h1B5: mem_out = 27'b000000000000000000000000000;
11'h1B6: mem_out = 27'b000000000000000000000000000;
11'h1B7: mem_out = 27'b000001000000000000000000000;
11'h1B8: mem_out = 27'b000001000000000000000000000;
11'h1B9: mem_out = 27'b000001000000000000000000000;
11'h1BA: mem_out = 27'b000001000000000000000000000;
11'h1BB: mem_out = 27'b000000000000000000000000000;
11'h1BC: mem_out = 27'b000000000000000000000000000;
11'h1BD: mem_out = 27'b000000000000000000000000000;
11'h1BE: mem_out = 27'b000000000000000000000000000;
11'h1BF: mem_out = 27'b000000000000000000000000000;
11'h1C0: mem_out = 27'b000000000000000000000000000;
11'h1C1: mem_out = 27'b000000000000000000000000000;
11'h1C2: mem_out = 27'b000000000000000000000000000;
11'h1C3: mem_out = 27'b000000000000000000000000000;
11'h1C4: mem_out = 27'b000000000000000000000000000;
11'h1C5: mem_out = 27'b000001000000000000000000000;
11'h1C6: mem_out = 27'b000001000000000000000000000;
11'h1C7: mem_out = 27'b000001000000000000000000000;
11'h1C8: mem_out = 27'b000001000000000000000000000;
11'h1C9: mem_out = 27'b000001000000000000000000000;
11'h1CA: mem_out = 27'b000001000000000000000000000;
11'h1CB: mem_out = 27'b000001000000000000000000000;
11'h1CC: mem_out = 27'b000001000000000000000000000;
11'h1CD: mem_out = 27'b000001000000000000000000000;
11'h1CE: mem_out = 27'b000001000000000000000000000;
11'h1CF: mem_out = 27'b000001000000000000000000000;
11'h1D0: mem_out = 27'b000001000000000000000000000;
```

```
11'h1D1: mem_out = 27'b000001000000000000000000000;
11'h1D2: mem_out = 27'b000001000000000000000000000;
11'h1D3: mem_out = 27'b000001000000000000000000000;
11'h1D4: mem_out = 27'b000001000000000000000000000;
11'h1D5: mem_out = 27'b000001000000000000000000000;
11'h1D6: mem_out = 27'b000001000000000000000000000;
11'h1D7: mem_out = 27'b000001000000000000000000000;
11'h1D8: mem_out = 27'b000001000000000000000000000;
11'h1D9: mem_out = 27'b000001000000000000000000000;
11'h1DA: mem_out = 27'b000001000000000000000000000;
11'h1DB: mem_out = 27'b000000000000000000000000000;
11'h1DC: mem_out = 27'b000000000000000000000000000;
11'h1DD: mem_out = 27'b000000000000000000000000000;
11'h1DE: mem_out = 27'b000000000000000000000000000;
11'h1DF: mem_out = 27'b000000000000000000000000000;
11'h1E0: mem_out = 27'b000000000000000000000000000;
11'h1E1: mem_out = 27'b000000000000000000000000000;
11'h1E2: mem_out = 27'b000000000000000000000000000;
11'h1E3: mem_out = 27'b000000000000000000000000000;
11'h1E4: mem_out = 27'b000000000000000000000000000;
11'h1E5: mem_out = 27'b000001000000000000000000000;
11'h1E6: mem_out = 27'b000001000000000000000000000;
11'h1E7: mem_out = 27'b000001000000000000000000000;
11'h1E8: mem_out = 27'b000001000000000000000000000;
11'h1E9: mem_out = 27'b000001000000000000000000000;
11'h1EA: mem_out = 27'b000001000000000000000000000;
11'h1EB: mem_out = 27'b000001000000000000000000000;
11'h1EC: mem_out = 27'b000001000000000000000000000;
11'h1ED: mem_out = 27'b000001000000000000000000000;
11'h1EE: mem_out = 27'b000001000000000000000000000;
11'h1EF: mem_out = 27'b000001000000000000000000000;
11'h1F0: mem_out = 27'b000001000000000000000000000;
11'h1F1: mem_out = 27'b000001000000000000000000000;
11'h1F2: mem_out = 27'b000001000000000000000000000;
11'h1F3: mem_out = 27'b000001000000000000000000000;
11'h1F4: mem_out = 27'b000001000000000000000000000;
11'h1F5: mem_out = 27'b000001000000000000000000000;
11'h1F6: mem_out = 27'b000001000000000000000000000;
11'h1F7: mem_out = 27'b000001000000000000000000000;
11'h1F8: mem_out = 27'b000001000000000000000000000;
11'h1F9: mem_out = 27'b000001000000000000000000000;
11'h1FA: mem_out = 27'b000001000000000000000000000;
11'h1FB: mem_out = 27'b000000000000000000000000000;
11'h1FC: mem_out = 27'b000000000000000000000000000;
```

```
11'h1FD: mem_out = 27'b000000000000000000000000000;
11'h1FE: mem_out = 27'b000000000000000000000000000;
11'h1FF: mem_out = 27'b000000000000000000000000000;
11'h200: mem_out = 27'b000000000000000000000000000;
11'h201: mem_out = 27'b000000000000000000000000000;
11'h202: mem_out = 27'b000000000000000000000000000;
11'h203: mem_out = 27'b000000000000000000000000000;
11'h204: mem_out = 27'b000000000000000000000000000;
11'h205: mem_out = 27'b000001000000000000000000000;
11'h206: mem_out = 27'b000001000000000000000000000;
11'h207: mem_out = 27'b000001000000000000000000000;
11'h208: mem_out = 27'b000001000000000000000000000;
11'h209: mem_out = 27'b000000000000000000000000000;
11'h20A: mem_out = 27'b000000000000000000000000000;
11'h20B: mem_out = 27'b000000000000000000000000000;
11'h20C: mem_out = 27'b000000000000000000000000000;
11'h20D: mem_out = 27'b000000000000000000000000000;
11'h20E: mem_out = 27'b000000000000000000000000000;
11'h20F: mem_out = 27'b000000000000000000000000000;
11'h210: mem_out = 27'b000000000000000000000000000;
11'h211: mem_out = 27'b000000000000000000000000000;
11'h212: mem_out = 27'b000000000000000000000000000;
11'h213: mem_out = 27'b000000000000000000000000000;
11'h214: mem_out = 27'b000000000000000000000000000;
11'h215: mem_out = 27'b000000000000000000000000000;
11'h216: mem_out = 27'b000000000000000000000000000;
11'h217: mem_out = 27'b000001000000000000000000000;
11'h218: mem_out = 27'b000001000000000000000000000;
11'h219: mem_out = 27'b000001000000000000000000000;
11'h21A: mem_out = 27'b000001000000000000000000000;
11'h21B: mem_out = 27'b000000000000000000000000000;
11'h21C: mem_out = 27'b000000000000000000000000000;
11'h21D: mem_out = 27'b000000000000000000000000000;
11'h21E: mem_out = 27'b000000000000000000000000000;
11'h21F: mem_out = 27'b000000000000000000000000000;
11'h220: mem_out = 27'b000000000000000000000000000;
11'h221: mem_out = 27'b000000000000000000000000000;
11'h222: mem_out = 27'b000000000000000000000000000;
11'h223: mem_out = 27'b000000000000000000000000000;
11'h224: mem_out = 27'b000000000000000000000000000;
11'h225: mem_out = 27'b000001000000000000000000000;
11'h226: mem_out = 27'b000001000000000000000000000;
11'h227: mem_out = 27'b000001000000000000000000000;
11'h228: mem_out = 27'b000001000000000000000000000;
```

```
11'h229: mem_out = 27'b000000000000000000000000000;
11'h22A: mem_out = 27'b000000000000000000000000000;
11'h22B: mem_out = 27'b000000000000000000000000000;
11'h22C: mem_out = 27'b000000000000000000000000000;
11'h22D: mem_out = 27'b000000000000000000000000000;
11'h22E: mem_out = 27'b000000000000000000000000000;
11'h22F: mem_out = 27'b000000000000000000000000000;
11'h230: mem_out = 27'b000000000000000000000000000;
11'h231: mem_out = 27'b000000000000000000000000000;
11'h232: mem_out = 27'b000000000000000000000000000;
11'h233: mem_out = 27'b000000000000000000000000000;
11'h234: mem_out = 27'b000000000000000000000000000;
11'h235: mem_out = 27'b000000000000000000000000000;
11'h236: mem_out = 27'b000000000000000000000000000;
11'h237: mem_out = 27'b000001000000000000000000000;
11'h238: mem_out = 27'b000001000000000000000000000;
11'h239: mem_out = 27'b000001000000000000000000000;
11'h23A: mem_out = 27'b000001000000000000000000000;
11'h23B: mem_out = 27'b000000000000000000000000000;
11'h23C: mem_out = 27'b000000000000000000000000000;
11'h23D: mem_out = 27'b000000000000000000000000000;
11'h23E: mem_out = 27'b000000000000000000000000000;
11'h23F: mem_out = 27'b000000000000000000000000000;
11'h240: mem_out = 27'b000000000000000000000000000;
11'h241: mem_out = 27'b000000000000000000000000000;
11'h242: mem_out = 27'b000000000000000000000000000;
11'h243: mem_out = 27'b000000000000000000000000000;
11'h244: mem_out = 27'b000000000000000000000000000;
11'h245: mem_out = 27'b000001000000000000000000000;
11'h246: mem_out = 27'b000001000000000000000000000;
11'h247: mem_out = 27'b000001000000000000000000000;
11'h248: mem_out = 27'b000001000000000000000000000;
11'h249: mem_out = 27'b000000000000000000000000000;
11'h24A: mem_out = 27'b000000000000000000000000000;
11'h24B: mem_out = 27'b000000000000000000000000000;
11'h24C: mem_out = 27'b000000000000000000000000000;
11'h24D: mem_out = 27'b000000000000000000000000000;
11'h24E: mem_out = 27'b000000000000000000000000000;
11'h24F: mem_out = 27'b000000000000000000000000000;
11'h250: mem_out = 27'b000000000000000000000000000;
11'h251: mem_out = 27'b000000000000000000000000000;
11'h252: mem_out = 27'b000000000000000000000000000;
11'h253: mem_out = 27'b000000000000000000000000000;
11'h254: mem_out = 27'b000000000000000000000000000;
```

```
11'h255: mem_out = 27'b000000000000000000000000000;
11'h256: mem_out = 27'b000000000000000000000000000;
11'h257: mem_out = 27'b000001000000000000000000000;
11'h258: mem_out = 27'b000001000000000000000000000;
11'h259: mem_out = 27'b000001000000000000000000000;
11'h25A: mem_out = 27'b000001000000000000000000000;
11'h25B: mem_out = 27'b000000000000000000000000000;
11'h25C: mem_out = 27'b000000000000000000000000000;
11'h25D: mem_out = 27'b000000000000000000000000000;
11'h25E: mem_out = 27'b000000000000000000000000000;
11'h25F: mem_out = 27'b000000000000000000000000000;
11'h260: mem_out = 27'b000000000000000000000000000;
11'h261: mem_out = 27'b000000000000000000000000000;
11'h262: mem_out = 27'b000000000000000000000000000;
11'h263: mem_out = 27'b000000000000000000000000000;
11'h264: mem_out = 27'b000000000000000000000000000;
11'h265: mem_out = 27'b000001000000000000000000000;
11'h266: mem_out = 27'b000001000000000000000000000;
11'h267: mem_out = 27'b000001000000000000000000000;
11'h268: mem_out = 27'b000001000000000000000000000;
11'h269: mem_out = 27'b000000000000000000000000000;
11'h26A: mem_out = 27'b000000000000000000000000000;
11'h26B: mem_out = 27'b000000000000000000000000000;
11'h26C: mem_out = 27'b000000000000000000000000000;
11'h26D: mem_out = 27'b000000000000000000000000000;
11'h26E: mem_out = 27'b000000000000000000000000000;
11'h26F: mem_out = 27'b000000000000000000000000000;
11'h270: mem_out = 27'b000000000000000000000000000;
11'h271: mem_out = 27'b000000000000000000000000000;
11'h272: mem_out = 27'b000000000000000000000000000;
11'h273: mem_out = 27'b000000000000000000000000000;
11'h274: mem_out = 27'b000000000000000000000000000;
11'h275: mem_out = 27'b000000000000000000000000000;
11'h276: mem_out = 27'b000000000000000000000000000;
11'h277: mem_out = 27'b000001000000000000000000000;
11'h278: mem_out = 27'b000001000000000000000000000;
11'h279: mem_out = 27'b000001000000000000000000000;
11'h27A: mem_out = 27'b000001000000000000000000000;
11'h27B: mem_out = 27'b000000000000000000000000000;
11'h27C: mem_out = 27'b000000000000000000000000000;
11'h27D: mem_out = 27'b000000000000000000000000000;
11'h27E: mem_out = 27'b000000000000000000000000000;
11'h27F: mem_out = 27'b000000000000000000000000000;
11'h280: mem_out = 27'b000000000000000000000000000;
```

```
11'h281: mem_out = 27'b000000000000000000000000000;
11'h282: mem_out = 27'b000000000000000000000000000;
11'h283: mem_out = 27'b000000000000000000000000000;
11'h284: mem_out = 27'b000000000000000000000000000;
11'h285: mem_out = 27'b000001000000000000000000000;
11'h286: mem_out = 27'b000001000000000000000000000;
11'h287: mem_out = 27'b000001000000000000000000000;
11'h288: mem_out = 27'b000001000000000000000000000;
11'h289: mem_out = 27'b000000000000000000000000000;
11'h28A: mem_out = 27'b000000000000000000000000000;
11'h28B: mem_out = 27'b000000000000000000000000000;
11'h28C: mem_out = 27'b000000000000000000000000000;
11'h28D: mem_out = 27'b000000000000000000000000000;
11'h28E: mem_out = 27'b000000000000000000000000000;
11'h28F: mem_out = 27'b000000000000000000000000000;
11'h290: mem_out = 27'b000000000000000000000000000;
11'h291: mem_out = 27'b000000000000000000000000000;
11'h292: mem_out = 27'b000000000000000000000000000;
11'h293: mem_out = 27'b000000000000000000000000000;
11'h294: mem_out = 27'b000000000000000000000000000;
11'h295: mem_out = 27'b000000000000000000000000000;
11'h296: mem_out = 27'b000000000000000000000000000;
11'h297: mem_out = 27'b000001000000000000000000000;
11'h298: mem_out = 27'b000001000000000000000000000;
11'h299: mem_out = 27'b000001000000000000000000000;
11'h29A: mem_out = 27'b000001000000000000000000000;
11'h29B: mem_out = 27'b000000000000000000000000000;
11'h29C: mem_out = 27'b000000000000000000000000000;
11'h29D: mem_out = 27'b000000000000000000000000000;
11'h29E: mem_out = 27'b000000000000000000000000000;
11'h29F: mem_out = 27'b000000000000000000000000000;
11'h2A0: mem_out = 27'b000000000000000000000000000;
11'h2A1: mem_out = 27'b000000000000000000000000000;
11'h2A2: mem_out = 27'b000000000000000000000000000;
11'h2A3: mem_out = 27'b000000000000000000000000000;
11'h2A4: mem_out = 27'b000000000000000000000000000;
11'h2A5: mem_out = 27'b000001000000000000000000000;
11'h2A6: mem_out = 27'b000001000000000000000000000;
11'h2A7: mem_out = 27'b000001000000000000000000000;
11'h2A8: mem_out = 27'b000001000000000000000000000;
11'h2A9: mem_out = 27'b000000000000000000000000000;
11'h2AA: mem_out = 27'b000000000000000000000000000;
11'h2AB: mem_out = 27'b000000000000000000000000000;
11'h2AC: mem_out = 27'b000000000000000000000000000;
```

```
11'h2AD: mem_out = 27'b000000000000000000000000000;
11'h2AE: mem_out = 27'b000000000000000000000000000;
11'h2AF: mem_out = 27'b000000000000000000000000000;
11'h2B0: mem_out = 27'b000000000000000000000000000;
11'h2B1: mem_out = 27'b000000000000000000000000000;
11'h2B2: mem_out = 27'b000000000000000000000000000;
11'h2B3: mem_out = 27'b000000000000000000000000000;
11'h2B4: mem_out = 27'b000000000000000000000000000;
11'h2B5: mem_out = 27'b000000000000000000000000000;
11'h2B6: mem_out = 27'b000000000000000000000000000;
11'h2B7: mem_out = 27'b000001000000000000000000000;
11'h2B8: mem_out = 27'b000001000000000000000000000;
11'h2B9: mem_out = 27'b000001000000000000000000000;
11'h2BA: mem_out = 27'b000001000000000000000000000;
11'h2BB: mem_out = 27'b000000000000000000000000000;
11'h2BC: mem_out = 27'b000000000000000000000000000;
11'h2BD: mem_out = 27'b000000000000000000000000000;
11'h2BE: mem_out = 27'b000000000000000000000000000;
11'h2BF: mem_out = 27'b000000000000000000000000000;
11'h2C0: mem_out = 27'b000000000000000000000000000;
11'h2C1: mem_out = 27'b000000000000000000000000000;
11'h2C2: mem_out = 27'b000000000000000000000000000;
11'h2C3: mem_out = 27'b000000000000000000000000000;
11'h2C4: mem_out = 27'b000000000000000000000000000;
11'h2C5: mem_out = 27'b000001000000000000000000000;
11'h2C6: mem_out = 27'b000001000000000000000000000;
11'h2C7: mem_out = 27'b000001000000000000000000000;
11'h2C8: mem_out = 27'b000001000000000000000000000;
11'h2C9: mem_out = 27'b000000000000000000000000000;
11'h2CA: mem_out = 27'b000000000000000000000000000;
11'h2CB: mem_out = 27'b000000000000000000000000000;
11'h2CC: mem_out = 27'b000000000000000000000000000;
11'h2CD: mem_out = 27'b000000000000000000000000000;
11'h2CE: mem_out = 27'b000000000000000000000000000;
11'h2CF: mem_out = 27'b000000000000000000000000000;
11'h2D0: mem_out = 27'b000000000000000000000000000;
11'h2D1: mem_out = 27'b000000000000000000000000000;
11'h2D2: mem_out = 27'b000000000000000000000000000;
11'h2D3: mem_out = 27'b000000000000000000000000000;
11'h2D4: mem_out = 27'b000000000000000000000000000;
11'h2D5: mem_out = 27'b000000000000000000000000000;
11'h2D6: mem_out = 27'b000000000000000000000000000;
11'h2D7: mem_out = 27'b000001000000000000000000000;
11'h2D8: mem_out = 27'b000001000000000000000000000;
```

```
11'h2D9: mem_out = 27'b000001000000000000000000000;
11'h2DA: mem_out = 27'b000001000000000000000000000;
11'h2DB: mem_out = 27'b000000000000000000000000000;
11'h2DC: mem_out = 27'b000000000000000000000000000;
11'h2DD: mem_out = 27'b000000000000000000000000000;
11'h2DE: mem_out = 27'b000000000000000000000000000;
11'h2DF: mem_out = 27'b000000000000000000000000000;
11'h2E0: mem_out = 27'b000000000000000000000000000;
11'h2E1: mem_out = 27'b000000000000000000000000000;
11'h2E2: mem_out = 27'b000000000000000000000000000;
11'h2E3: mem_out = 27'b000000000000000000000000000;
11'h2E4: mem_out = 27'b000000000000000000000000000;
11'h2E5: mem_out = 27'b000001000000000000000000000;
11'h2E6: mem_out = 27'b000001000000000000000000000;
11'h2E7: mem_out = 27'b000001000000000000000000000;
11'h2E8: mem_out = 27'b000001000000000000000000000;
11'h2E9: mem_out = 27'b000000000000000000000000000;
11'h2EA: mem_out = 27'b000000000000000000000000000;
11'h2EB: mem_out = 27'b000000000000000000000000000;
11'h2EC: mem_out = 27'b000000000000000000000000000;
11'h2ED: mem_out = 27'b000000000000000000000000000;
11'h2EE: mem_out = 27'b000000000000000000000000000;
11'h2EF: mem_out = 27'b000000000000000000000000000;
11'h2F0: mem_out = 27'b000000000000000000000000000;
11'h2F1: mem_out = 27'b000000000000000000000000000;
11'h2F2: mem_out = 27'b000000000000000000000000000;
11'h2F3: mem_out = 27'b000000000000000000000000000;
11'h2F4: mem_out = 27'b000000000000000000000000000;
11'h2F5: mem_out = 27'b000000000000000000000000000;
11'h2F6: mem_out = 27'b000000000000000000000000000;
11'h2F7: mem_out = 27'b000001000000000000000000000;
11'h2F8: mem_out = 27'b000001000000000000000000000;
11'h2F9: mem_out = 27'b000001000000000000000000000;
11'h2FA: mem_out = 27'b000001000000000000000000000;
11'h2FB: mem_out = 27'b000000000000000000000000000;
11'h2FC: mem_out = 27'b000000000000000000000000000;
11'h2FD: mem_out = 27'b000000000000000000000000000;
11'h2FE: mem_out = 27'b000000000000000000000000000;
11'h2FF: mem_out = 27'b000000000000000000000000000;
11'h300: mem_out = 27'b000000000000000000000000000;
11'h301: mem_out = 27'b000000000000000000000000000;
11'h302: mem_out = 27'b000000000000000000000000000;
11'h303: mem_out = 27'b000000000000000000000000000;
11'h304: mem_out = 27'b000000000000000000000000000;
```

```
11'h305: mem_out = 27'b000001000000000000000000000;
11'h306: mem_out = 27'b000001000000000000000000000;
11'h307: mem_out = 27'b000001000000000000000000000;
11'h308: mem_out = 27'b000001000000000000000000000;
11'h309: mem_out = 27'b000000000000000000000000000;
11'h30A: mem_out = 27'b000000000000000000000000000;
11'h30B: mem_out = 27'b000000000000000000000000000;
11'h30C: mem_out = 27'b000000000000000000000000000;
11'h30D: mem_out = 27'b000000000000000000000000000;
11'h30E: mem_out = 27'b000000000000000000000000000;
11'h30F: mem_out = 27'b000000000000000000000000000;
11'h310: mem_out = 27'b000000000000000000000000000;
11'h311: mem_out = 27'b000000000000000000000000000;
11'h312: mem_out = 27'b000000000000000000000000000;
11'h313: mem_out = 27'b000000000000000000000000000;
11'h314: mem_out = 27'b000000000000000000000000000;
11'h315: mem_out = 27'b000000000000000000000000000;
11'h316: mem_out = 27'b000000000000000000000000000;
11'h317: mem_out = 27'b000001000000000000000000000;
11'h318: mem_out = 27'b000001000000000000000000000;
11'h319: mem_out = 27'b000001000000000000000000000;
11'h31A: mem_out = 27'b000001000000000000000000000;
11'h31B: mem_out = 27'b000000000000000000000000000;
11'h31C: mem_out = 27'b000000000000000000000000000;
11'h31D: mem_out = 27'b000000000000000000000000000;
11'h31E: mem_out = 27'b000000000000000000000000000;
11'h31F: mem_out = 27'b000000000000000000000000000;
11'h320: mem_out = 27'b000000000000000000000000000;
11'h321: mem_out = 27'b000000000000000000000000000;
11'h322: mem_out = 27'b000000000000000000000000000;
11'h323: mem_out = 27'b000000000000000000000000000;
11'h324: mem_out = 27'b000000000000000000000000000;
11'h325: mem_out = 27'b000001000000000000000000000;
11'h326: mem_out = 27'b000001000000000000000000000;
11'h327: mem_out = 27'b000001000000000000000000000;
11'h328: mem_out = 27'b000001000000000000000000000;
11'h329: mem_out = 27'b000000000000000000000000000;
11'h32A: mem_out = 27'b000000000000000000000000000;
11'h32B: mem_out = 27'b000000000000000000000000000;
11'h32C: mem_out = 27'b000000000000000000000000000;
11'h32D: mem_out = 27'b000000000000000000000000000;
11'h32E: mem_out = 27'b000000000000000000000000000;
11'h32F: mem_out = 27'b000000000000000000000000000;
11'h330: mem_out = 27'b000000000000000000000000000;
```

```
11'h331: mem_out = 27'b000000000000000000000000000;
11'h332: mem_out = 27'b000000000000000000000000000;
11'h333: mem_out = 27'b000000000000000000000000000;
11'h334: mem_out = 27'b000000000000000000000000000;
11'h335: mem_out = 27'b000000000000000000000000000;
11'h336: mem_out = 27'b000000000000000000000000000;
11'h337: mem_out = 27'b000001000000000000000000000;
11'h338: mem_out = 27'b000001000000000000000000000;
11'h339: mem_out = 27'b000001000000000000000000000;
11'h33A: mem_out = 27'b000001000000000000000000000;
11'h33B: mem_out = 27'b000000000000000000000000000;
11'h33C: mem_out = 27'b000000000000000000000000000;
11'h33D: mem_out = 27'b000000000000000000000000000;
11'h33E: mem_out = 27'b000000000000000000000000000;
11'h33F: mem_out = 27'b000000000000000000000000000;
11'h340: mem_out = 27'b000000000000000000000000000;
11'h341: mem_out = 27'b000000000000000000000000000;
11'h342: mem_out = 27'b000000000000000000000000000;
11'h343: mem_out = 27'b000000000000000000000000000;
11'h344: mem_out = 27'b000000000000000000000000000;
11'h345: mem_out = 27'b000001000000000000000000000;
11'h346: mem_out = 27'b000001000000000000000000000;
11'h347: mem_out = 27'b000001000000000000000000000;
11'h348: mem_out = 27'b000001000000000000000000000;
11'h349: mem_out = 27'b000000000000000000000000000;
11'h34A: mem_out = 27'b000000000000000000000000000;
11'h34B: mem_out = 27'b000000000000000000000000000;
11'h34C: mem_out = 27'b000000000000000000000000000;
11'h34D: mem_out = 27'b000000000000000000000000000;
11'h34E: mem_out = 27'b000000000000000000000000000;
11'h34F: mem_out = 27'b000000000000000000000000000;
11'h350: mem_out = 27'b000000000000000000000000000;
11'h351: mem_out = 27'b000000000000000000000000000;
11'h352: mem_out = 27'b000000000000000000000000000;
11'h353: mem_out = 27'b000000000000000000000000000;
11'h354: mem_out = 27'b000000000000000000000000000;
11'h355: mem_out = 27'b000000000000000000000000000;
11'h356: mem_out = 27'b000000000000000000000000000;
11'h357: mem_out = 27'b000001000000000000000000000;
11'h358: mem_out = 27'b000001000000000000000000000;
11'h359: mem_out = 27'b000001000000000000000000000;
11'h35A: mem_out = 27'b000001000000000000000000000;
11'h35B: mem_out = 27'b000000000000000000000000000;
11'h35C: mem_out = 27'b000000000000000000000000000;
```

```
11'h35D: mem_out = 27'b000000000000000000000000000;
11'h35E: mem_out = 27'b000000000000000000000000000;
11'h35F: mem_out = 27'b000000000000000000000000000;
11'h360: mem_out = 27'b000000000000000000000000000;
11'h361: mem_out = 27'b000000000000000000000000000;
11'h362: mem_out = 27'b000000000000000000000000000;
11'h363: mem_out = 27'b000000000000000000000000000;
11'h364: mem_out = 27'b000000000000000000000000000;
11'h365: mem_out = 27'b000001000000000000000000000;
11'h366: mem_out = 27'b000001000000000000000000000;
11'h367: mem_out = 27'b000001000000000000000000000;
11'h368: mem_out = 27'b000001000000000000000000000;
11'h369: mem_out = 27'b000000000000000000000000000;
11'h36A: mem_out = 27'b000000000000000000000000000;
11'h36B: mem_out = 27'b000000000000000000000000000;
11'h36C: mem_out = 27'b000000000000000000000000000;
11'h36D: mem_out = 27'b000000000000000000000000000;
11'h36E: mem_out = 27'b000000000000000000000000000;
11'h36F: mem_out = 27'b000000000000000000000000000;
11'h370: mem_out = 27'b000000000000000000000000000;
11'h371: mem_out = 27'b000000000000000000000000000;
11'h372: mem_out = 27'b000000000000000000000000000;
11'h373: mem_out = 27'b000000000000000000000000000;
11'h374: mem_out = 27'b000000000000000000000000000;
11'h375: mem_out = 27'b000000000000000000000000000;
11'h376: mem_out = 27'b000000000000000000000000000;
11'h377: mem_out = 27'b000001000000000000000000000;
11'h378: mem_out = 27'b000001000000000000000000000;
11'h379: mem_out = 27'b000001000000000000000000000;
11'h37A: mem_out = 27'b000001000000000000000000000;
11'h37B: mem_out = 27'b000000000000000000000000000;
11'h37C: mem_out = 27'b000000000000000000000000000;
11'h37D: mem_out = 27'b000000000000000000000000000;
11'h37E: mem_out = 27'b000000000000000000000000000;
11'h37F: mem_out = 27'b000000000000000000000000000;
11'h380: mem_out = 27'b000000000000000000000000000;
11'h381: mem_out = 27'b000000000000000000000000000;
11'h382: mem_out = 27'b000001000000000000000000000;
11'h383: mem_out = 27'b000001000000000000000000000;
11'h384: mem_out = 27'b000001000000000000000000000;
11'h385: mem_out = 27'b000001000000000000000000000;
11'h386: mem_out = 27'b000001000000000000000000000;
11'h387: mem_out = 27'b000001000000000000000000000;
11'h388: mem_out = 27'b000001000000000000000000000;
```

```
11'h389: mem_out = 27'b000001000000000000000000000;
11'h38A: mem_out = 27'b000001000000000000000000000;
11'h38B: mem_out = 27'b000001000000000000000000000;
11'h38C: mem_out = 27'b000000000000000000000000000;
11'h38D: mem_out = 27'b000000000000000000000000000;
11'h38E: mem_out = 27'b000000000000000000000000000;
11'h38F: mem_out = 27'b000000000000000000000000000;
11'h390: mem_out = 27'b000000000000000000000000000;
11'h391: mem_out = 27'b000000000000000000000000000;
11'h392: mem_out = 27'b000000000000000000000000000;
11'h393: mem_out = 27'b000000000000000000000000000;
11'h394: mem_out = 27'b000001000000000000000000000;
11'h395: mem_out = 27'b000001000000000000000000000;
11'h396: mem_out = 27'b000001000000000000000000000;
11'h397: mem_out = 27'b000001000000000000000000000;
11'h398: mem_out = 27'b000001000000000000000000000;
11'h399: mem_out = 27'b000001000000000000000000000;
11'h39A: mem_out = 27'b000001000000000000000000000;
11'h39B: mem_out = 27'b000001000000000000000000000;
11'h39C: mem_out = 27'b000001000000000000000000000;
11'h39D: mem_out = 27'b000001000000000000000000000;
11'h39E: mem_out = 27'b000000000000000000000000000;
11'h39F: mem_out = 27'b000000000000000000000000000;
11'h3A0: mem_out = 27'b000000000000000000000000000;
11'h3A1: mem_out = 27'b000000000000000000000000000;
11'h3A2: mem_out = 27'b000001000000000000000000000;
11'h3A3: mem_out = 27'b000001000000000000000000000;
11'h3A4: mem_out = 27'b000001000000000000000000000;
11'h3A5: mem_out = 27'b000001000000000000000000000;
11'h3A6: mem_out = 27'b000001000000000000000000000;
11'h3A7: mem_out = 27'b000001000000000000000000000;
11'h3A8: mem_out = 27'b000001000000000000000000000;
11'h3A9: mem_out = 27'b000001000000000000000000000;
11'h3AA: mem_out = 27'b000001000000000000000000000;
11'h3AB: mem_out = 27'b000001000000000000000000000;
11'h3AC: mem_out = 27'b000001000000000000000000000;
11'h3AD: mem_out = 27'b000000000000000000000000000;
11'h3AE: mem_out = 27'b000000000000000000000000000;
11'h3AF: mem_out = 27'b000000000000000000000000000;
11'h3B0: mem_out = 27'b000000000000000000000000000;
11'h3B1: mem_out = 27'b000000000000000000000000000;
11'h3B2: mem_out = 27'b000000000000000000000000000;
11'h3B3: mem_out = 27'b000001000000000000000000000;
11'h3B4: mem_out = 27'b000001000000000000000000000;
```

```
11'h3B5: mem_out = 27'b000001000000000000000000000;
11'h3B6: mem_out = 27'b000001000000000000000000000;
11'h3B7: mem_out = 27'b000001000000000000000000000;
11'h3B8: mem_out = 27'b000001000000000000000000000;
11'h3B9: mem_out = 27'b000001000000000000000000000;
11'h3BA: mem_out = 27'b000001000000000000000000000;
11'h3BB: mem_out = 27'b000001000000000000000000000;
11'h3BC: mem_out = 27'b000001000000000000000000000;
11'h3BD: mem_out = 27'b000001000000000000000000000;
11'h3BE: mem_out = 27'b000000000000000000000000000;
11'h3BF: mem_out = 27'b000000000000000000000000000;
11'h3C0: mem_out = 27'b000000000000000000000000000;
11'h3C1: mem_out = 27'b000000000000000000000000000;
11'h3C2: mem_out = 27'b000000000000000000000000000;
11'h3C3: mem_out = 27'b000000000000000000000000000;
11'h3C4: mem_out = 27'b000000000000000000000000000;
11'h3C5: mem_out = 27'b000000000000000000000000000;
11'h3C6: mem_out = 27'b000000000000000000000000000;
11'h3C7: mem_out = 27'b000000000000000000000000000;
11'h3C8: mem_out = 27'b000000000000000000000000000;
11'h3C9: mem_out = 27'b000000000000000000000000000;
11'h3CA: mem_out = 27'b000000000000000000000000000;
11'h3CB: mem_out = 27'b000000000000000000000000000;
11'h3CC: mem_out = 27'b000000000000000000000000000;
11'h3CD: mem_out = 27'b000000000000000000000000000;
11'h3CE: mem_out = 27'b000000000000000000000000000;
11'h3CF: mem_out = 27'b000000000000000000000000000;
11'h3D0: mem_out = 27'b000000000000000000000000000;
11'h3D1: mem_out = 27'b000000000000000000000000000;
11'h3D2: mem_out = 27'b000000000000000000000000000;
11'h3D3: mem_out = 27'b000000000000000000000000000;
11'h3D4: mem_out = 27'b000000000000000000000000000;
11'h3D5: mem_out = 27'b000000000000000000000000000;
11'h3D6: mem_out = 27'b000000000000000000000000000;
11'h3D7: mem_out = 27'b000000000000000000000000000;
11'h3D8: mem_out = 27'b000000000000000000000000000;
11'h3D9: mem_out = 27'b000000000000000000000000000;
11'h3DA: mem_out = 27'b000000000000000000000000000;
11'h3DB: mem_out = 27'b000000000000000000000000000;
11'h3DC: mem_out = 27'b000000000000000000000000000;
11'h3DD: mem_out = 27'b000000000000000000000000000;
11'h3DE: mem_out = 27'b000000000000000000000000000;
11'h3DF: mem_out = 27'b000000000000000000000000000;
11'h3E0: mem_out = 27'b000000000000000000000000000;
```

```
                11'h3E1: mem_out = 27'b000000000000000000000000000;
                11'h3E2: mem_out = 27'b000000000000000000000000000;
                11'h3E3: mem_out = 27'b000000000000000000000000000;
                11'h3E4: mem_out = 27'b000000000000000000000000000;
                11'h3E5: mem_out = 27'b000000000000000000000000000;
                11'h3E6: mem_out = 27'b000000000000000000000000000;
                11'h3E7: mem_out = 27'b000000000000000000000000000;
                11'h3E8: mem_out = 27'b000000000000000000000000000;
                11'h3E9: mem_out = 27'b000000000000000000000000000;
                11'h3EA: mem_out = 27'b000000000000000000000000000;
                11'h3EB: mem_out = 27'b000000000000000000000000000;
                11'h3EC: mem_out = 27'b000000000000000000000000000;
                11'h3ED: mem_out = 27'b000000000000000000000000000;
                11'h3EE: mem_out = 27'b000000000000000000000000000;
                11'h3EF: mem_out = 27'b000000000000000000000000000;
                11'h3F0: mem_out = 27'b000000000000000000000000000;
                11'h3F1: mem_out = 27'b000000000000000000000000000;
                11'h3F2: mem_out = 27'b000000000000000000000000000;
                11'h3F3: mem_out = 27'b000000000000000000000000000;
                11'h3F4: mem_out = 27'b000000000000000000000000000;
                11'h3F5: mem_out = 27'b000000000000000000000000000;
                11'h3F6: mem_out = 27'b000000000000000000000000000;
                11'h3F7: mem_out = 27'b000000000000000000000000000;
                11'h3F8: mem_out = 27'b000000000000000000000000000;
                11'h3F9: mem_out = 27'b000000000000000000000000000;
                11'h3FA: mem_out = 27'b000000000000000000000000000;
                11'h3FB: mem_out = 27'b000000000000000000000000000;
                11'h3FC: mem_out = 27'b000000000000000000000000000;
                11'h3FD: mem_out = 27'b000000000000000000000000000;
                11'h3FE: mem_out = 27'b000000000000000000000000000;
                11'h3FF: mem_out = 27'b000000000000000000000000000;

                endcase

        end
endmodule
*/
module sync_rom_zero(clk,address,mem_out);
        input           clk;
        input    [10:0] address;
        output   [26:0] mem_out;
        reg signed [26:0] mem_out;
        always@(posedge clk) begin
                case(address)
```

```
11'h0: mem_out = 27'b000000000000000000000000000;
11'h1: mem_out = 27'b000000000000000000000000000;
11'h2: mem_out = 27'b000000000000000000000000000;
11'h3: mem_out = 27'b000000000000000000000000000;
11'h4: mem_out = 27'b000000000000000000000000000;
11'h5: mem_out = 27'b000000000000000000000000000;
11'h6: mem_out = 27'b000000000000000000000000000;
11'h7: mem_out = 27'b000000000000000000000000000;
11'h8: mem_out = 27'b000000000000000000000000000;
11'h9: mem_out = 27'b000000000000000000000000000;
11'hA: mem_out = 27'b000000000000000000000000000;
11'hB: mem_out = 27'b000000000000000000000000000;
11'hC: mem_out = 27'b000000000000000000000000000;
11'hD: mem_out = 27'b000000000000000000000000000;
11'hE: mem_out = 27'b000000000000000000000000000;
11'hF: mem_out = 27'b000000000000000000000000000;
11'h10: mem_out = 27'b000000000000000000000000000;
11'h11: mem_out = 27'b000000000000000000000000000;
11'h12: mem_out = 27'b000000000000000000000000000;
11'h13: mem_out = 27'b000000000000000000000000000;
11'h14: mem_out = 27'b000000000000000000000000000;
11'h15: mem_out = 27'b000000000000000000000000000;
11'h16: mem_out = 27'b000000000000000000000000000;
11'h17: mem_out = 27'b000000000000000000000000000;
11'h18: mem_out = 27'b000000000000000000000000000;
11'h19: mem_out = 27'b000000000000000000000000000;
11'h1A: mem_out = 27'b000000000000000000000000000;
11'h1B: mem_out = 27'b000000000000000000000000000;
11'h1C: mem_out = 27'b000000000000000000000000000;
11'h1D: mem_out = 27'b000000000000000000000000000;
11'h1E: mem_out = 27'b000000000000000000000000000;
11'h1F: mem_out = 27'b000000000000000000000000000;
11'h20: mem_out = 27'b000000000000000000000000000;
11'h21: mem_out = 27'b000000000000000000000000000;
11'h22: mem_out = 27'b000000000000000000000000000;
11'h23: mem_out = 27'b000000000000000000000000000;
11'h24: mem_out = 27'b000000000000000000000000000;
11'h25: mem_out = 27'b000000000000000000000000000;
11'h26: mem_out = 27'b000000000000000000000000000;
11'h27: mem_out = 27'b000000000000000000000000000;
11'h28: mem_out = 27'b000000000000000000000000000;
11'h29: mem_out = 27'b000000000000000000000000000;
11'h2A: mem_out = 27'b000000000000000000000000000;
11'h2B: mem_out = 27'b000000000000000000000000000;
```

```
11'h2C: mem_out = 27'b000000000000000000000000000;
11'h2D: mem_out = 27'b000000000000000000000000000;
11'h2E: mem_out = 27'b000000000000000000000000000;
11'h2F: mem_out = 27'b000000000000000000000000000;
11'h30: mem_out = 27'b000000000000000000000000000;
11'h31: mem_out = 27'b000000000000000000000000000;
11'h32: mem_out = 27'b000000000000000000000000000;
11'h33: mem_out = 27'b000000000000000000000000000;
11'h34: mem_out = 27'b000000000000000000000000000;
11'h35: mem_out = 27'b000000000000000000000000000;
11'h36: mem_out = 27'b000000000000000000000000000;
11'h37: mem_out = 27'b000000000000000000000000000;
11'h38: mem_out = 27'b000000000000000000000000000;
11'h39: mem_out = 27'b000000000000000000000000000;
11'h3A: mem_out = 27'b000000000000000000000000000;
11'h3B: mem_out = 27'b000000000000000000000000000;
11'h3C: mem_out = 27'b000000000000000000000000000;
11'h3D: mem_out = 27'b000000000000000000000000000;
11'h3E: mem_out = 27'b000000000000000000000000000;
11'h3F: mem_out = 27'b000000000000000000000000000;
11'h40: mem_out = 27'b000000000000000000000000000;
11'h41: mem_out = 27'b000000000000000000000000000;
11'h42: mem_out = 27'b000000000000000000000000000;
11'h43: mem_out = 27'b000000000000000000000000000;
11'h44: mem_out = 27'b000000000000000000000000000;
11'h45: mem_out = 27'b000000000000000000000000000;
11'h46: mem_out = 27'b000000000000000000000000000;
11'h47: mem_out = 27'b000000000000000000000000000;
11'h48: mem_out = 27'b000000000000000000000000000;
11'h49: mem_out = 27'b000000000000000000000000000;
11'h4A: mem_out = 27'b000000000000000000000000000;
11'h4B: mem_out = 27'b000000000000000000000000000;
11'h4C: mem_out = 27'b000000000000000000000000000;
11'h4D: mem_out = 27'b000000000000000000000000000;
11'h4E: mem_out = 27'b000000000000000000000000000;
11'h4F: mem_out = 27'b000000000000000000000000000;
11'h50: mem_out = 27'b000000000000000000000000000;
11'h51: mem_out = 27'b000000000000000000000000000;
11'h52: mem_out = 27'b000000000000000000000000000;
11'h53: mem_out = 27'b000000000000000000000000000;
11'h54: mem_out = 27'b000000000000000000000000000;
11'h55: mem_out = 27'b000000000000000000000000000;
11'h56: mem_out = 27'b000000000000000000000000000;
11'h57: mem_out = 27'b000000000000000000000000000;
```

```
11'h58: mem_out = 27'b000000000000000000000000000;
11'h59: mem_out = 27'b000000000000000000000000000;
11'h5A: mem_out = 27'b000000000000000000000000000;
11'h5B: mem_out = 27'b000000000000000000000000000;
11'h5C: mem_out = 27'b000000000000000000000000000;
11'h5D: mem_out = 27'b000000000000000000000000000;
11'h5E: mem_out = 27'b000000000000000000000000000;
11'h5F: mem_out = 27'b000000000000000000000000000;
11'h60: mem_out = 27'b000000000000000000000000000;
11'h61: mem_out = 27'b000000000000000000000000000;
11'h62: mem_out = 27'b000000000000000000000000000;
11'h63: mem_out = 27'b000000000000000000000000000;
11'h64: mem_out = 27'b000000000000000000000000000;
11'h65: mem_out = 27'b000000000000000000000000000;
11'h66: mem_out = 27'b000000000000000000000000000;
11'h67: mem_out = 27'b000000000000000000000000000;
11'h68: mem_out = 27'b000000000000000000000000000;
11'h69: mem_out = 27'b000000000000000000000000000;
11'h6A: mem_out = 27'b000000000000000000000000000;
11'h6B: mem_out = 27'b000000000000000000000000000;
11'h6C: mem_out = 27'b000000000000000000000000000;
11'h6D: mem_out = 27'b000000000000000000000000000;
11'h6E: mem_out = 27'b000000000000000000000000000;
11'h6F: mem_out = 27'b000000000000000000000000000;
11'h70: mem_out = 27'b000000000000000000000000000;
11'h71: mem_out = 27'b000000000000000000000000000;
11'h72: mem_out = 27'b000000000000000000000000000;
11'h73: mem_out = 27'b000000000000000000000000000;
11'h74: mem_out = 27'b000000000000000000000000000;
11'h75: mem_out = 27'b000000000000000000000000000;
11'h76: mem_out = 27'b000000000000000000000000000;
11'h77: mem_out = 27'b000000000000000000000000000;
11'h78: mem_out = 27'b000000000000000000000000000;
11'h79: mem_out = 27'b000000000000000000000000000;
11'h7A: mem_out = 27'b000000000000000000000000000;
11'h7B: mem_out = 27'b000000000000000000000000000;
11'h7C: mem_out = 27'b000000000000000000000000000;
11'h7D: mem_out = 27'b000000000000000000000000000;
11'h7E: mem_out = 27'b000000000000000000000000000;
11'h7F: mem_out = 27'b000000000000000000000000000;
11'h80: mem_out = 27'b000000000000000000000000000;
11'h81: mem_out = 27'b000000000000000000000000000;
11'h82: mem_out = 27'b000000000000000000000000000;
11'h83: mem_out = 27'b000000000000000000000000000;
```

```
11'h84: mem_out = 27'b000000000000000000000000000;
11'h85: mem_out = 27'b000000000000000000000000000;
11'h86: mem_out = 27'b000000000000000000000000000;
11'h87: mem_out = 27'b000000000000000000000000000;
11'h88: mem_out = 27'b000000000000000000000000000;
11'h89: mem_out = 27'b000000000000000000000000000;
11'h8A: mem_out = 27'b000000000000000000000000000;
11'h8B: mem_out = 27'b000000000000000000000000000;
11'h8C: mem_out = 27'b000000000000000000000000000;
11'h8D: mem_out = 27'b000000000000000000000000000;
11'h8E: mem_out = 27'b000000000000000000000000000;
11'h8F: mem_out = 27'b000000000000000000000000000;
11'h90: mem_out = 27'b000000000000000000000000000;
11'h91: mem_out = 27'b000000000000000000000000000;
11'h92: mem_out = 27'b000000000000000000000000000;
11'h93: mem_out = 27'b000000000000000000000000000;
11'h94: mem_out = 27'b000000000000000000000000000;
11'h95: mem_out = 27'b000000000000000000000000000;
11'h96: mem_out = 27'b000000000000000000000000000;
11'h97: mem_out = 27'b000000000000000000000000000;
11'h98: mem_out = 27'b000000000000000000000000000;
11'h99: mem_out = 27'b000000000000000000000000000;
11'h9A: mem_out = 27'b000000000000000000000000000;
11'h9B: mem_out = 27'b000000000000000000000000000;
11'h9C: mem_out = 27'b000000000000000000000000000;
11'h9D: mem_out = 27'b000000000000000000000000000;
11'h9E: mem_out = 27'b000000000000000000000000000;
11'h9F: mem_out = 27'b000000000000000000000000000;
11'hA0: mem_out = 27'b000000000000000000000000000;
11'hA1: mem_out = 27'b000000000000000000000000000;
11'hA2: mem_out = 27'b000000000000000000000000000;
11'hA3: mem_out = 27'b000000000000000000000000000;
11'hA4: mem_out = 27'b000000000000000000000000000;
11'hA5: mem_out = 27'b000000000000000000000000000;
11'hA6: mem_out = 27'b000000000000000000000000000;
11'hA7: mem_out = 27'b000000000000000000000000000;
11'hA8: mem_out = 27'b000000000000000000000000000;
11'hA9: mem_out = 27'b000000000000000000000000000;
11'hAA: mem_out = 27'b000000000000000000000000000;
11'hAB: mem_out = 27'b000000000000000000000000000;
11'hAC: mem_out = 27'b000000000000000000000000000;
11'hAD: mem_out = 27'b000000000000000000000000000;
11'hAE: mem_out = 27'b000000000000000000000000000;
11'hAF: mem_out = 27'b000000000000000000000000000;
```

```
11'hB0: mem_out = 27'b000000000000000000000000000;
11'hB1: mem_out = 27'b000000000000000000000000000;
11'hB2: mem_out = 27'b000000000000000000000000000;
11'hB3: mem_out = 27'b000000000000000000000000000;
11'hB4: mem_out = 27'b000000000000000000000000000;
11'hB5: mem_out = 27'b000000000000000000000000000;
11'hB6: mem_out = 27'b000000000000000000000000000;
11'hB7: mem_out = 27'b000000000000000000000000000;
11'hB8: mem_out = 27'b000000000000000000000000000;
11'hB9: mem_out = 27'b000000000000000000000000000;
11'hBA: mem_out = 27'b000000000000000000000000000;
11'hBB: mem_out = 27'b000000000000000000000000000;
11'hBC: mem_out = 27'b000000000000000000000000000;
11'hBD: mem_out = 27'b000000000000000000000000000;
11'hBE: mem_out = 27'b000000000000000000000000000;
11'hBF: mem_out = 27'b000000000000000000000000000;
11'hC0: mem_out = 27'b000000000000000000000000000;
11'hC1: mem_out = 27'b000000000000000000000000000;
11'hC2: mem_out = 27'b000000000000000000000000000;
11'hC3: mem_out = 27'b000000000000000000000000000;
11'hC4: mem_out = 27'b000000000000000000000000000;
11'hC5: mem_out = 27'b000000000000000000000000000;
11'hC6: mem_out = 27'b000000000000000000000000000;
11'hC7: mem_out = 27'b000000000000000000000000000;
11'hC8: mem_out = 27'b000000000000000000000000000;
11'hC9: mem_out = 27'b000000000000000000000000000;
11'hCA: mem_out = 27'b000000000000000000000000000;
11'hCB: mem_out = 27'b000000000000000000000000000;
11'hCC: mem_out = 27'b000000000000000000000000000;
11'hCD: mem_out = 27'b000000000000000000000000000;
11'hCE: mem_out = 27'b000000000000000000000000000;
11'hCF: mem_out = 27'b000000000000000000000000000;
11'hD0: mem_out = 27'b000000000000000000000000000;
11'hD1: mem_out = 27'b000000000000000000000000000;
11'hD2: mem_out = 27'b000000000000000000000000000;
11'hD3: mem_out = 27'b000000000000000000000000000;
11'hD4: mem_out = 27'b000000000000000000000000000;
11'hD5: mem_out = 27'b000000000000000000000000000;
11'hD6: mem_out = 27'b000000000000000000000000000;
11'hD7: mem_out = 27'b000000000000000000000000000;
11'hD8: mem_out = 27'b000000000000000000000000000;
11'hD9: mem_out = 27'b000000000000000000000000000;
11'hDA: mem_out = 27'b000000000000000000000000000;
11'hDB: mem_out = 27'b000000000000000000000000000;
```

```
11'hDC: mem_out = 27'b000000000000000000000000000;
11'hDD: mem_out = 27'b000000000000000000000000000;
11'hDE: mem_out = 27'b000000000000000000000000000;
11'hDF: mem_out = 27'b000000000000000000000000000;
11'hE0: mem_out = 27'b000000000000000000000000000;
11'hE1: mem_out = 27'b000000000000000000000000000;
11'hE2: mem_out = 27'b000000000000000000000000000;
11'hE3: mem_out = 27'b000000000000000000000000000;
11'hE4: mem_out = 27'b000000000000000000000000000;
11'hE5: mem_out = 27'b000000000000000000000000000;
11'hE6: mem_out = 27'b000000000000000000000000000;
11'hE7: mem_out = 27'b000000000000000000000000000;
11'hE8: mem_out = 27'b000000000000000000000000000;
11'hE9: mem_out = 27'b000000000000000000000000000;
11'hEA: mem_out = 27'b000000000000000000000000000;
11'hEB: mem_out = 27'b000000000000000000000000000;
11'hEC: mem_out = 27'b000000000000000000000000000;
11'hED: mem_out = 27'b000000000000000000000000000;
11'hEE: mem_out = 27'b000000000000000000000000000;
11'hEF: mem_out = 27'b000000000000000000000000000;
11'hF0: mem_out = 27'b000000000000000000000000000;
11'hF1: mem_out = 27'b000000000000000000000000000;
11'hF2: mem_out = 27'b000000000000000000000000000;
11'hF3: mem_out = 27'b000000000000000000000000000;
11'hF4: mem_out = 27'b000000000000000000000000000;
11'hF5: mem_out = 27'b000000000000000000000000000;
11'hF6: mem_out = 27'b000000000000000000000000000;
11'hF7: mem_out = 27'b000000000000000000000000000;
11'hF8: mem_out = 27'b000000000000000000000000000;
11'hF9: mem_out = 27'b000000000000000000000000000;
11'hFA: mem_out = 27'b000000000000000000000000000;
11'hFB: mem_out = 27'b000000000000000000000000000;
11'hFC: mem_out = 27'b000000000000000000000000000;
11'hFD: mem_out = 27'b000000000000000000000000000;
11'hFE: mem_out = 27'b000000000000000000000000000;
11'hFF: mem_out = 27'b000000000000000000000000000;
11'h100: mem_out = 27'b000000000000000000000000000;
11'h101: mem_out = 27'b000000000000000000000000000;
11'h102: mem_out = 27'b000000000000000000000000000;
11'h103: mem_out = 27'b000000000000000000000000000;
11'h104: mem_out = 27'b000000000000000000000000000;
11'h105: mem_out = 27'b000000000000000000000000000;
11'h106: mem_out = 27'b000000000000000000000000000;
11'h107: mem_out = 27'b000000000000000000000000000;
```

```
11'h108: mem_out = 27'b000000000000000000000000000;
11'h109: mem_out = 27'b000000000000000000000000000;
11'h10A: mem_out = 27'b000000000000000000000000000;
11'h10B: mem_out = 27'b000000000000000000000000000;
11'h10C: mem_out = 27'b000000000000000000000000000;
11'h10D: mem_out = 27'b000000000000000000000000000;
11'h10E: mem_out = 27'b000000000000000000000000000;
11'h10F: mem_out = 27'b000000000000000000000000000;
11'h110: mem_out = 27'b000000000000000000000000000;
11'h111: mem_out = 27'b000000000000000000000000000;
11'h112: mem_out = 27'b000000000000000000000000000;
11'h113: mem_out = 27'b000000000000000000000000000;
11'h114: mem_out = 27'b000000000000000000000000000;
11'h115: mem_out = 27'b000000000000000000000000000;
11'h116: mem_out = 27'b000000000000000000000000000;
11'h117: mem_out = 27'b000000000000000000000000000;
11'h118: mem_out = 27'b000000000000000000000000000;
11'h119: mem_out = 27'b000000000000000000000000000;
11'h11A: mem_out = 27'b000000000000000000000000000;
11'h11B: mem_out = 27'b000000000000000000000000000;
11'h11C: mem_out = 27'b000000000000000000000000000;
11'h11D: mem_out = 27'b000000000000000000000000000;
11'h11E: mem_out = 27'b000000000000000000000000000;
11'h11F: mem_out = 27'b000000000000000000000000000;
11'h120: mem_out = 27'b000000000000000000000000000;
11'h121: mem_out = 27'b000000000000000000000000000;
11'h122: mem_out = 27'b000000000000000000000000000;
11'h123: mem_out = 27'b000000000000000000000000000;
11'h124: mem_out = 27'b000000000000000000000000000;
11'h125: mem_out = 27'b000000000000000000000000000;
11'h126: mem_out = 27'b000000000000000000000000000;
11'h127: mem_out = 27'b000000000000000000000000000;
11'h128: mem_out = 27'b000000000000000000000000000;
11'h129: mem_out = 27'b000000000000000000000000000;
11'h12A: mem_out = 27'b000000000000000000000000000;
11'h12B: mem_out = 27'b000000000000000000000000000;
11'h12C: mem_out = 27'b000000000000000000000000000;
11'h12D: mem_out = 27'b000000000000000000000000000;
11'h12E: mem_out = 27'b000000000000000000000000000;
11'h12F: mem_out = 27'b000000000000000000000000000;
11'h130: mem_out = 27'b000000000000000000000000000;
11'h131: mem_out = 27'b000000000000000000000000000;
11'h132: mem_out = 27'b000000000000000000000000000;
11'h133: mem_out = 27'b000000000000000000000000000;
```

```
11'h134: mem_out = 27'b000000000000000000000000000;
11'h135: mem_out = 27'b000000000000000000000000000;
11'h136: mem_out = 27'b000000000000000000000000000;
11'h137: mem_out = 27'b000000000000000000000000000;
11'h138: mem_out = 27'b000000000000000000000000000;
11'h139: mem_out = 27'b000000000000000000000000000;
11'h13A: mem_out = 27'b000000000000000000000000000;
11'h13B: mem_out = 27'b000000000000000000000000000;
11'h13C: mem_out = 27'b000000000000000000000000000;
11'h13D: mem_out = 27'b000000000000000000000000000;
11'h13E: mem_out = 27'b000000000000000000000000000;
11'h13F: mem_out = 27'b000000000000000000000000000;
11'h140: mem_out = 27'b000000000000000000000000000;
11'h141: mem_out = 27'b000000000000000000000000000;
11'h142: mem_out = 27'b000000000000000000000000000;
11'h143: mem_out = 27'b000000000000000000000000000;
11'h144: mem_out = 27'b000000000000000000000000000;
11'h145: mem_out = 27'b000000000000000000000000000;
11'h146: mem_out = 27'b000000000000000000000000000;
11'h147: mem_out = 27'b000000000000000000000000000;
11'h148: mem_out = 27'b000000000000000000000000000;
11'h149: mem_out = 27'b000000000000000000000000000;
11'h14A: mem_out = 27'b000000000000000000000000000;
11'h14B: mem_out = 27'b000000000000000000000000000;
11'h14C: mem_out = 27'b000000000000000000000000000;
11'h14D: mem_out = 27'b000000000000000000000000000;
11'h14E: mem_out = 27'b000000000000000000000000000;
11'h14F: mem_out = 27'b000000000000000000000000000;
11'h150: mem_out = 27'b000000000000000000000000000;
11'h151: mem_out = 27'b000000000000000000000000000;
11'h152: mem_out = 27'b000000000000000000000000000;
11'h153: mem_out = 27'b000000000000000000000000000;
11'h154: mem_out = 27'b000000000000000000000000000;
11'h155: mem_out = 27'b000000000000000000000000000;
11'h156: mem_out = 27'b000000000000000000000000000;
11'h157: mem_out = 27'b000000000000000000000000000;
11'h158: mem_out = 27'b000000000000000000000000000;
11'h159: mem_out = 27'b000000000000000000000000000;
11'h15A: mem_out = 27'b000000000000000000000000000;
11'h15B: mem_out = 27'b000000000000000000000000000;
11'h15C: mem_out = 27'b000000000000000000000000000;
11'h15D: mem_out = 27'b000000000000000000000000000;
11'h15E: mem_out = 27'b000000000000000000000000000;
11'h15F: mem_out = 27'b000000000000000000000000000;
```

```
11'h160: mem_out = 27'b000000000000000000000000000;
11'h161: mem_out = 27'b000000000000000000000000000;
11'h162: mem_out = 27'b000000000000000000000000000;
11'h163: mem_out = 27'b000000000000000000000000000;
11'h164: mem_out = 27'b000000000000000000000000000;
11'h165: mem_out = 27'b000000000000000000000000000;
11'h166: mem_out = 27'b000000000000000000000000000;
11'h167: mem_out = 27'b000000000000000000000000000;
11'h168: mem_out = 27'b000000000000000000000000000;
11'h169: mem_out = 27'b000000000000000000000000000;
11'h16A: mem_out = 27'b000000000000000000000000000;
11'h16B: mem_out = 27'b000000000000000000000000000;
11'h16C: mem_out = 27'b000000000000000000000000000;
11'h16D: mem_out = 27'b000000000000000000000000000;
11'h16E: mem_out = 27'b000000000000000000000000000;
11'h16F: mem_out = 27'b000000000000000000000000000;
11'h170: mem_out = 27'b000000000000000000000000000;
11'h171: mem_out = 27'b000000000000000000000000000;
11'h172: mem_out = 27'b000000000000000000000000000;
11'h173: mem_out = 27'b000000000000000000000000000;
11'h174: mem_out = 27'b000000000000000000000000000;
11'h175: mem_out = 27'b000000000000000000000000000;
11'h176: mem_out = 27'b000000000000000000000000000;
11'h177: mem_out = 27'b000000000000000000000000000;
11'h178: mem_out = 27'b000000000000000000000000000;
11'h179: mem_out = 27'b000000000000000000000000000;
11'h17A: mem_out = 27'b000000000000000000000000000;
11'h17B: mem_out = 27'b000000000000000000000000000;
11'h17C: mem_out = 27'b000000000000000000000000000;
11'h17D: mem_out = 27'b000000000000000000000000000;
11'h17E: mem_out = 27'b000000000000000000000000000;
11'h17F: mem_out = 27'b000000000000000000000000000;
11'h180: mem_out = 27'b000000000000000000000000000;
11'h181: mem_out = 27'b000000000000000000000000000;
11'h182: mem_out = 27'b000000000000000000000000000;
11'h183: mem_out = 27'b000000000000000000000000000;
11'h184: mem_out = 27'b000000000000000000000000000;
11'h185: mem_out = 27'b000000000000000000000000000;
11'h186: mem_out = 27'b000000000000000000000000000;
11'h187: mem_out = 27'b000000000000000000000000000;
11'h188: mem_out = 27'b000000000000000000000000000;
11'h189: mem_out = 27'b000000000000000000000000000;
11'h18A: mem_out = 27'b000000000000000000000000000;
11'h18B: mem_out = 27'b000000000000000000000000000;
```

```
11'h18C: mem_out = 27'b000000000000000000000000000;
11'h18D: mem_out = 27'b000000000000000000000000000;
11'h18E: mem_out = 27'b000000000000000000000000000;
11'h18F: mem_out = 27'b000000000000000000000000000;
11'h190: mem_out = 27'b000000000000000000000000000;
11'h191: mem_out = 27'b000000000000000000000000000;
11'h192: mem_out = 27'b000000000000000000000000000;
11'h193: mem_out = 27'b000000000000000000000000000;
11'h194: mem_out = 27'b000000000000000000000000000;
11'h195: mem_out = 27'b000000000000000000000000000;
11'h196: mem_out = 27'b000000000000000000000000000;
11'h197: mem_out = 27'b000000000000000000000000000;
11'h198: mem_out = 27'b000000000000000000000000000;
11'h199: mem_out = 27'b000000000000000000000000000;
11'h19A: mem_out = 27'b000000000000000000000000000;
11'h19B: mem_out = 27'b000000000000000000000000000;
11'h19C: mem_out = 27'b000000000000000000000000000;
11'h19D: mem_out = 27'b000000000000000000000000000;
11'h19E: mem_out = 27'b000000000000000000000000000;
11'h19F: mem_out = 27'b000000000000000000000000000;
11'h1A0: mem_out = 27'b000000000000000000000000000;
11'h1A1: mem_out = 27'b000000000000000000000000000;
11'h1A2: mem_out = 27'b000000000000000000000000000;
11'h1A3: mem_out = 27'b000000000000000000000000000;
11'h1A4: mem_out = 27'b000000000000000000000000000;
11'h1A5: mem_out = 27'b000000000000000000000000000;
11'h1A6: mem_out = 27'b000000000000000000000000000;
11'h1A7: mem_out = 27'b000000000000000000000000000;
11'h1A8: mem_out = 27'b000000000000000000000000000;
11'h1A9: mem_out = 27'b000000000000000000000000000;
11'h1AA: mem_out = 27'b000000000000000000000000000;
11'h1AB: mem_out = 27'b000000000000000000000000000;
11'h1AC: mem_out = 27'b000000000000000000000000000;
11'h1AD: mem_out = 27'b000000000000000000000000000;
11'h1AE: mem_out = 27'b000000000000000000000000000;
11'h1AF: mem_out = 27'b000000000000000000000000000;
11'h1B0: mem_out = 27'b000000000000000000000000000;
11'h1B1: mem_out = 27'b000000000000000000000000000;
11'h1B2: mem_out = 27'b000000000000000000000000000;
11'h1B3: mem_out = 27'b000000000000000000000000000;
11'h1B4: mem_out = 27'b000000000000000000000000000;
11'h1B5: mem_out = 27'b000000000000000000000000000;
11'h1B6: mem_out = 27'b000000000000000000000000000;
11'h1B7: mem_out = 27'b000000000000000000000000000;
```

```
11'h1B8: mem_out = 27'b000000000000000000000000000;
11'h1B9: mem_out = 27'b000000000000000000000000000;
11'h1BA: mem_out = 27'b000000000000000000000000000;
11'h1BB: mem_out = 27'b000000000000000000000000000;
11'h1BC: mem_out = 27'b000000000000000000000000000;
11'h1BD: mem_out = 27'b000000000000000000000000000;
11'h1BE: mem_out = 27'b000000000000000000000000000;
11'h1BF: mem_out = 27'b000000000000000000000000000;
11'h1C0: mem_out = 27'b000000000000000000000000000;
11'h1C1: mem_out = 27'b000000000000000000000000000;
11'h1C2: mem_out = 27'b000000000000000000000000000;
11'h1C3: mem_out = 27'b000000000000000000000000000;
11'h1C4: mem_out = 27'b000000000000000000000000000;
11'h1C5: mem_out = 27'b000000000000000000000000000;
11'h1C6: mem_out = 27'b000000000000000000000000000;
11'h1C7: mem_out = 27'b000000000000000000000000000;
11'h1C8: mem_out = 27'b000000000000000000000000000;
11'h1C9: mem_out = 27'b000000000000000000000000000;
11'h1CA: mem_out = 27'b000000000000000000000000000;
11'h1CB: mem_out = 27'b000000000000000000000000000;
11'h1CC: mem_out = 27'b000000000000000000000000000;
11'h1CD: mem_out = 27'b000000000000000000000000000;
11'h1CE: mem_out = 27'b000000000000000000000000000;
11'h1CF: mem_out = 27'b000000000000000000000000000;
11'h1D0: mem_out = 27'b000000000000000000000000000;
11'h1D1: mem_out = 27'b000000000000000000000000000;
11'h1D2: mem_out = 27'b000000000000000000000000000;
11'h1D3: mem_out = 27'b000000000000000000000000000;
11'h1D4: mem_out = 27'b000000000000000000000000000;
11'h1D5: mem_out = 27'b000000000000000000000000000;
11'h1D6: mem_out = 27'b000000000000000000000000000;
11'h1D7: mem_out = 27'b000000000000000000000000000;
11'h1D8: mem_out = 27'b000000000000000000000000000;
11'h1D9: mem_out = 27'b000000000000000000000000000;
11'h1DA: mem_out = 27'b000000000000000000000000000;
11'h1DB: mem_out = 27'b000000000000000000000000000;
11'h1DC: mem_out = 27'b000000000000000000000000000;
11'h1DD: mem_out = 27'b000000000000000000000000000;
11'h1DE: mem_out = 27'b000000000000000000000000000;
11'h1DF: mem_out = 27'b000000000000000000000000000;
11'h1E0: mem_out = 27'b000000000000000000000000000;
11'h1E1: mem_out = 27'b000000000000000000000000000;
11'h1E2: mem_out = 27'b000000000000000000000000000;
11'h1E3: mem_out = 27'b000000000000000000000000000;
```

```
11'h1E4: mem_out = 27'b000000000000000000000000000;
11'h1E5: mem_out = 27'b000000000000000000000000000;
11'h1E6: mem_out = 27'b000000000000000000000000000;
11'h1E7: mem_out = 27'b000000000000000000000000000;
11'h1E8: mem_out = 27'b000000000000000000000000000;
11'h1E9: mem_out = 27'b000000000000000000000000000;
11'h1EA: mem_out = 27'b000000000000000000000000000;
11'h1EB: mem_out = 27'b000000000000000000000000000;
11'h1EC: mem_out = 27'b000000000000000000000000000;
11'h1ED: mem_out = 27'b000000000000000000000000000;
11'h1EE: mem_out = 27'b000000000000000000000000000;
11'h1EF: mem_out = 27'b000000000000000000000000000;
11'h1F0: mem_out = 27'b000000000000000000000000000;
11'h1F1: mem_out = 27'b000000000000000000000000000;
11'h1F2: mem_out = 27'b000000000000000000000000000;
11'h1F3: mem_out = 27'b000000000000000000000000000;
11'h1F4: mem_out = 27'b000000000000000000000000000;
11'h1F5: mem_out = 27'b000000000000000000000000000;
11'h1F6: mem_out = 27'b000000000000000000000000000;
11'h1F7: mem_out = 27'b000000000000000000000000000;
11'h1F8: mem_out = 27'b000000000000000000000000000;
11'h1F9: mem_out = 27'b000000000000000000000000000;
11'h1FA: mem_out = 27'b000000000000000000000000000;
11'h1FB: mem_out = 27'b000000000000000000000000000;
11'h1FC: mem_out = 27'b000000000000000000000000000;
11'h1FD: mem_out = 27'b000000000000000000000000000;
11'h1FE: mem_out = 27'b000000000000000000000000000;
11'h1FF: mem_out = 27'b000000000000000000000000000;
11'h200: mem_out = 27'b000000000000000000000000000;
11'h201: mem_out = 27'b000000000000000000000000000;
11'h202: mem_out = 27'b000000000000000000000000000;
11'h203: mem_out = 27'b000000000000000000000000000;
11'h204: mem_out = 27'b000000000000000000000000000;
11'h205: mem_out = 27'b000000000000000000000000000;
11'h206: mem_out = 27'b000000000000000000000000000;
11'h207: mem_out = 27'b000000000000000000000000000;
11'h208: mem_out = 27'b000000000000000000000000000;
11'h209: mem_out = 27'b000000000000000000000000000;
11'h20A: mem_out = 27'b000000000000000000000000000;
11'h20B: mem_out = 27'b000000000000000000000000000;
11'h20C: mem_out = 27'b000000000000000000000000000;
11'h20D: mem_out = 27'b000000000000000000000000000;
11'h20E: mem_out = 27'b000000000000000000000000000;
11'h20F: mem_out = 27'b000000000000000000000000000;
```

```
11'h210: mem_out = 27'b000000000000000000000000000;
11'h211: mem_out = 27'b000000000000000000000000000;
11'h212: mem_out = 27'b000000000000000000000000000;
11'h213: mem_out = 27'b000000000000000000000000000;
11'h214: mem_out = 27'b000000000000000000000000000;
11'h215: mem_out = 27'b000000000000000000000000000;
11'h216: mem_out = 27'b000000000000000000000000000;
11'h217: mem_out = 27'b000000000000000000000000000;
11'h218: mem_out = 27'b000000000000000000000000000;
11'h219: mem_out = 27'b000000000000000000000000000;
11'h21A: mem_out = 27'b000000000000000000000000000;
11'h21B: mem_out = 27'b000000000000000000000000000;
11'h21C: mem_out = 27'b000000000000000000000000000;
11'h21D: mem_out = 27'b000000000000000000000000000;
11'h21E: mem_out = 27'b000000000000000000000000000;
11'h21F: mem_out = 27'b000000000000000000000000000;
11'h220: mem_out = 27'b000000000000000000000000000;
11'h221: mem_out = 27'b000000000000000000000000000;
11'h222: mem_out = 27'b000000000000000000000000000;
11'h223: mem_out = 27'b000000000000000000000000000;
11'h224: mem_out = 27'b000000000000000000000000000;
11'h225: mem_out = 27'b000000000000000000000000000;
11'h226: mem_out = 27'b000000000000000000000000000;
11'h227: mem_out = 27'b000000000000000000000000000;
11'h228: mem_out = 27'b000000000000000000000000000;
11'h229: mem_out = 27'b000000000000000000000000000;
11'h22A: mem_out = 27'b000000000000000000000000000;
11'h22B: mem_out = 27'b000000000000000000000000000;
11'h22C: mem_out = 27'b000000000000000000000000000;
11'h22D: mem_out = 27'b000000000000000000000000000;
11'h22E: mem_out = 27'b000000000000000000000000000;
11'h22F: mem_out = 27'b000000000000000000000000000;
11'h230: mem_out = 27'b000000000000000000000000000;
11'h231: mem_out = 27'b000000000000000000000000000;
11'h232: mem_out = 27'b000000000000000000000000000;
11'h233: mem_out = 27'b000000000000000000000000000;
11'h234: mem_out = 27'b000000000000000000000000000;
11'h235: mem_out = 27'b000000000000000000000000000;
11'h236: mem_out = 27'b000000000000000000000000000;
11'h237: mem_out = 27'b000000000000000000000000000;
11'h238: mem_out = 27'b000000000000000000000000000;
11'h239: mem_out = 27'b000000000000000000000000000;
11'h23A: mem_out = 27'b000000000000000000000000000;
11'h23B: mem_out = 27'b000000000000000000000000000;
```

```
11'h23C: mem_out = 27'b000000000000000000000000000;
11'h23D: mem_out = 27'b000000000000000000000000000;
11'h23E: mem_out = 27'b000000000000000000000000000;
11'h23F: mem_out = 27'b000000000000000000000000000;
11'h240: mem_out = 27'b000000000000000000000000000;
11'h241: mem_out = 27'b000000000000000000000000000;
11'h242: mem_out = 27'b000000000000000000000000000;
11'h243: mem_out = 27'b000000000000000000000000000;
11'h244: mem_out = 27'b000000000000000000000000000;
11'h245: mem_out = 27'b000000000000000000000000000;
11'h246: mem_out = 27'b000000000000000000000000000;
11'h247: mem_out = 27'b000000000000000000000000000;
11'h248: mem_out = 27'b000000000000000000000000000;
11'h249: mem_out = 27'b000000000000000000000000000;
11'h24A: mem_out = 27'b000000000000000000000000000;
11'h24B: mem_out = 27'b000000000000000000000000000;
11'h24C: mem_out = 27'b000000000000000000000000000;
11'h24D: mem_out = 27'b000000000000000000000000000;
11'h24E: mem_out = 27'b000000000000000000000000000;
11'h24F: mem_out = 27'b000000000000000000000000000;
11'h250: mem_out = 27'b000000000000000000000000000;
11'h251: mem_out = 27'b000000000000000000000000000;
11'h252: mem_out = 27'b000000000000000000000000000;
11'h253: mem_out = 27'b000000000000000000000000000;
11'h254: mem_out = 27'b000000000000000000000000000;
11'h255: mem_out = 27'b000000000000000000000000000;
11'h256: mem_out = 27'b000000000000000000000000000;
11'h257: mem_out = 27'b000000000000000000000000000;
11'h258: mem_out = 27'b000000000000000000000000000;
11'h259: mem_out = 27'b000000000000000000000000000;
11'h25A: mem_out = 27'b000000000000000000000000000;
11'h25B: mem_out = 27'b000000000000000000000000000;
11'h25C: mem_out = 27'b000000000000000000000000000;
11'h25D: mem_out = 27'b000000000000000000000000000;
11'h25E: mem_out = 27'b000000000000000000000000000;
11'h25F: mem_out = 27'b000000000000000000000000000;
11'h260: mem_out = 27'b000000000000000000000000000;
11'h261: mem_out = 27'b000000000000000000000000000;
11'h262: mem_out = 27'b000000000000000000000000000;
11'h263: mem_out = 27'b000000000000000000000000000;
11'h264: mem_out = 27'b000000000000000000000000000;
11'h265: mem_out = 27'b000000000000000000000000000;
11'h266: mem_out = 27'b000000000000000000000000000;
11'h267: mem_out = 27'b000000000000000000000000000;
```

```
11'h268: mem_out = 27'b000000000000000000000000000;
11'h269: mem_out = 27'b000000000000000000000000000;
11'h26A: mem_out = 27'b000000000000000000000000000;
11'h26B: mem_out = 27'b000000000000000000000000000;
11'h26C: mem_out = 27'b000000000000000000000000000;
11'h26D: mem_out = 27'b000000000000000000000000000;
11'h26E: mem_out = 27'b000000000000000000000000000;
11'h26F: mem_out = 27'b000000000000000000000000000;
11'h270: mem_out = 27'b000000000000000000000000000;
11'h271: mem_out = 27'b000000000000000000000000000;
11'h272: mem_out = 27'b000000000000000000000000000;
11'h273: mem_out = 27'b000000000000000000000000000;
11'h274: mem_out = 27'b000000000000000000000000000;
11'h275: mem_out = 27'b000000000000000000000000000;
11'h276: mem_out = 27'b000000000000000000000000000;
11'h277: mem_out = 27'b000000000000000000000000000;
11'h278: mem_out = 27'b000000000000000000000000000;
11'h279: mem_out = 27'b000000000000000000000000000;
11'h27A: mem_out = 27'b000000000000000000000000000;
11'h27B: mem_out = 27'b000000000000000000000000000;
11'h27C: mem_out = 27'b000000000000000000000000000;
11'h27D: mem_out = 27'b000000000000000000000000000;
11'h27E: mem_out = 27'b000000000000000000000000000;
11'h27F: mem_out = 27'b000000000000000000000000000;
11'h280: mem_out = 27'b000000000000000000000000000;
11'h281: mem_out = 27'b000000000000000000000000000;
11'h282: mem_out = 27'b000000000000000000000000000;
11'h283: mem_out = 27'b000000000000000000000000000;
11'h284: mem_out = 27'b000000000000000000000000000;
11'h285: mem_out = 27'b000000000000000000000000000;
11'h286: mem_out = 27'b000000000000000000000000000;
11'h287: mem_out = 27'b000000000000000000000000000;
11'h288: mem_out = 27'b000000000000000000000000000;
11'h289: mem_out = 27'b000000000000000000000000000;
11'h28A: mem_out = 27'b000000000000000000000000000;
11'h28B: mem_out = 27'b000000000000000000000000000;
11'h28C: mem_out = 27'b000000000000000000000000000;
11'h28D: mem_out = 27'b000000000000000000000000000;
11'h28E: mem_out = 27'b000000000000000000000000000;
11'h28F: mem_out = 27'b000000000000000000000000000;
11'h290: mem_out = 27'b000000000000000000000000000;
11'h291: mem_out = 27'b000000000000000000000000000;
11'h292: mem_out = 27'b000000000000000000000000000;
11'h293: mem_out = 27'b000000000000000000000000000;
```

```
11'h294: mem_out = 27'b000000000000000000000000000;
11'h295: mem_out = 27'b000000000000000000000000000;
11'h296: mem_out = 27'b000000000000000000000000000;
11'h297: mem_out = 27'b000000000000000000000000000;
11'h298: mem_out = 27'b000000000000000000000000000;
11'h299: mem_out = 27'b000000000000000000000000000;
11'h29A: mem_out = 27'b000000000000000000000000000;
11'h29B: mem_out = 27'b000000000000000000000000000;
11'h29C: mem_out = 27'b000000000000000000000000000;
11'h29D: mem_out = 27'b000000000000000000000000000;
11'h29E: mem_out = 27'b000000000000000000000000000;
11'h29F: mem_out = 27'b000000000000000000000000000;
11'h2A0: mem_out = 27'b000000000000000000000000000;
11'h2A1: mem_out = 27'b000000000000000000000000000;
11'h2A2: mem_out = 27'b000000000000000000000000000;
11'h2A3: mem_out = 27'b000000000000000000000000000;
11'h2A4: mem_out = 27'b000000000000000000000000000;
11'h2A5: mem_out = 27'b000000000000000000000000000;
11'h2A6: mem_out = 27'b000000000000000000000000000;
11'h2A7: mem_out = 27'b000000000000000000000000000;
11'h2A8: mem_out = 27'b000000000000000000000000000;
11'h2A9: mem_out = 27'b000000000000000000000000000;
11'h2AA: mem_out = 27'b000000000000000000000000000;
11'h2AB: mem_out = 27'b000000000000000000000000000;
11'h2AC: mem_out = 27'b000000000000000000000000000;
11'h2AD: mem_out = 27'b000000000000000000000000000;
11'h2AE: mem_out = 27'b000000000000000000000000000;
11'h2AF: mem_out = 27'b000000000000000000000000000;
11'h2B0: mem_out = 27'b000000000000000000000000000;
11'h2B1: mem_out = 27'b000000000000000000000000000;
11'h2B2: mem_out = 27'b000000000000000000000000000;
11'h2B3: mem_out = 27'b000000000000000000000000000;
11'h2B4: mem_out = 27'b000000000000000000000000000;
11'h2B5: mem_out = 27'b000000000000000000000000000;
11'h2B6: mem_out = 27'b000000000000000000000000000;
11'h2B7: mem_out = 27'b000000000000000000000000000;
11'h2B8: mem_out = 27'b000000000000000000000000000;
11'h2B9: mem_out = 27'b000000000000000000000000000;
11'h2BA: mem_out = 27'b000000000000000000000000000;
11'h2BB: mem_out = 27'b000000000000000000000000000;
11'h2BC: mem_out = 27'b000000000000000000000000000;
11'h2BD: mem_out = 27'b000000000000000000000000000;
11'h2BE: mem_out = 27'b000000000000000000000000000;
11'h2BF: mem_out = 27'b000000000000000000000000000;
```

```
11'h2C0: mem_out = 27'b000000000000000000000000000;
11'h2C1: mem_out = 27'b000000000000000000000000000;
11'h2C2: mem_out = 27'b000000000000000000000000000;
11'h2C3: mem_out = 27'b000000000000000000000000000;
11'h2C4: mem_out = 27'b000000000000000000000000000;
11'h2C5: mem_out = 27'b000000000000000000000000000;
11'h2C6: mem_out = 27'b000000000000000000000000000;
11'h2C7: mem_out = 27'b000000000000000000000000000;
11'h2C8: mem_out = 27'b000000000000000000000000000;
11'h2C9: mem_out = 27'b000000000000000000000000000;
11'h2CA: mem_out = 27'b000000000000000000000000000;
11'h2CB: mem_out = 27'b000000000000000000000000000;
11'h2CC: mem_out = 27'b000000000000000000000000000;
11'h2CD: mem_out = 27'b000000000000000000000000000;
11'h2CE: mem_out = 27'b000000000000000000000000000;
11'h2CF: mem_out = 27'b000000000000000000000000000;
11'h2D0: mem_out = 27'b000000000000000000000000000;
11'h2D1: mem_out = 27'b000000000000000000000000000;
11'h2D2: mem_out = 27'b000000000000000000000000000;
11'h2D3: mem_out = 27'b000000000000000000000000000;
11'h2D4: mem_out = 27'b000000000000000000000000000;
11'h2D5: mem_out = 27'b000000000000000000000000000;
11'h2D6: mem_out = 27'b000000000000000000000000000;
11'h2D7: mem_out = 27'b000000000000000000000000000;
11'h2D8: mem_out = 27'b000000000000000000000000000;
11'h2D9: mem_out = 27'b000000000000000000000000000;
11'h2DA: mem_out = 27'b000000000000000000000000000;
11'h2DB: mem_out = 27'b000000000000000000000000000;
11'h2DC: mem_out = 27'b000000000000000000000000000;
11'h2DD: mem_out = 27'b000000000000000000000000000;
11'h2DE: mem_out = 27'b000000000000000000000000000;
11'h2DF: mem_out = 27'b000000000000000000000000000;
11'h2E0: mem_out = 27'b000000000000000000000000000;
11'h2E1: mem_out = 27'b000000000000000000000000000;
11'h2E2: mem_out = 27'b000000000000000000000000000;
11'h2E3: mem_out = 27'b000000000000000000000000000;
11'h2E4: mem_out = 27'b000000000000000000000000000;
11'h2E5: mem_out = 27'b000000000000000000000000000;
11'h2E6: mem_out = 27'b000000000000000000000000000;
11'h2E7: mem_out = 27'b000000000000000000000000000;
11'h2E8: mem_out = 27'b000000000000000000000000000;
11'h2E9: mem_out = 27'b000000000000000000000000000;
11'h2EA: mem_out = 27'b000000000000000000000000000;
11'h2EB: mem_out = 27'b000000000000000000000000000;
```

```
11'h2EC: mem_out = 27'b000000000000000000000000000;
11'h2ED: mem_out = 27'b000000000000000000000000000;
11'h2EE: mem_out = 27'b000000000000000000000000000;
11'h2EF: mem_out = 27'b000000000000000000000000000;
11'h2F0: mem_out = 27'b000000000000000000000000000;
11'h2F1: mem_out = 27'b000000000000000000000000000;
11'h2F2: mem_out = 27'b000000000000000000000000000;
11'h2F3: mem_out = 27'b000000000000000000000000000;
11'h2F4: mem_out = 27'b000000000000000000000000000;
11'h2F5: mem_out = 27'b000000000000000000000000000;
11'h2F6: mem_out = 27'b000000000000000000000000000;
11'h2F7: mem_out = 27'b000000000000000000000000000;
11'h2F8: mem_out = 27'b000000000000000000000000000;
11'h2F9: mem_out = 27'b000000000000000000000000000;
11'h2FA: mem_out = 27'b000000000000000000000000000;
11'h2FB: mem_out = 27'b000000000000000000000000000;
11'h2FC: mem_out = 27'b000000000000000000000000000;
11'h2FD: mem_out = 27'b000000000000000000000000000;
11'h2FE: mem_out = 27'b000000000000000000000000000;
11'h2FF: mem_out = 27'b000000000000000000000000000;
11'h300: mem_out = 27'b000000000000000000000000000;
11'h301: mem_out = 27'b000000000000000000000000000;
11'h302: mem_out = 27'b000000000000000000000000000;
11'h303: mem_out = 27'b000000000000000000000000000;
11'h304: mem_out = 27'b000000000000000000000000000;
11'h305: mem_out = 27'b000000000000000000000000000;
11'h306: mem_out = 27'b000000000000000000000000000;
11'h307: mem_out = 27'b000000000000000000000000000;
11'h308: mem_out = 27'b000000000000000000000000000;
11'h309: mem_out = 27'b000000000000000000000000000;
11'h30A: mem_out = 27'b000000000000000000000000000;
11'h30B: mem_out = 27'b000000000000000000000000000;
11'h30C: mem_out = 27'b000000000000000000000000000;
11'h30D: mem_out = 27'b000000000000000000000000000;
11'h30E: mem_out = 27'b000000000000000000000000000;
11'h30F: mem_out = 27'b000000000000000000000000000;
11'h310: mem_out = 27'b000000000000000000000000000;
11'h311: mem_out = 27'b000000000000000000000000000;
11'h312: mem_out = 27'b000000000000000000000000000;
11'h313: mem_out = 27'b000000000000000000000000000;
11'h314: mem_out = 27'b000000000000000000000000000;
11'h315: mem_out = 27'b000000000000000000000000000;
11'h316: mem_out = 27'b000000000000000000000000000;
11'h317: mem_out = 27'b000000000000000000000000000;
```

```
11'h318: mem_out = 27'b000000000000000000000000000;
11'h319: mem_out = 27'b000000000000000000000000000;
11'h31A: mem_out = 27'b000000000000000000000000000;
11'h31B: mem_out = 27'b000000000000000000000000000;
11'h31C: mem_out = 27'b000000000000000000000000000;
11'h31D: mem_out = 27'b000000000000000000000000000;
11'h31E: mem_out = 27'b000000000000000000000000000;
11'h31F: mem_out = 27'b000000000000000000000000000;
11'h320: mem_out = 27'b000000000000000000000000000;
11'h321: mem_out = 27'b000000000000000000000000000;
11'h322: mem_out = 27'b000000000000000000000000000;
11'h323: mem_out = 27'b000000000000000000000000000;
11'h324: mem_out = 27'b000000000000000000000000000;
11'h325: mem_out = 27'b000000000000000000000000000;
11'h326: mem_out = 27'b000000000000000000000000000;
11'h327: mem_out = 27'b000000000000000000000000000;
11'h328: mem_out = 27'b000000000000000000000000000;
11'h329: mem_out = 27'b000000000000000000000000000;
11'h32A: mem_out = 27'b000000000000000000000000000;
11'h32B: mem_out = 27'b000000000000000000000000000;
11'h32C: mem_out = 27'b000000000000000000000000000;
11'h32D: mem_out = 27'b000000000000000000000000000;
11'h32E: mem_out = 27'b000000000000000000000000000;
11'h32F: mem_out = 27'b000000000000000000000000000;
11'h330: mem_out = 27'b000000000000000000000000000;
11'h331: mem_out = 27'b000000000000000000000000000;
11'h332: mem_out = 27'b000000000000000000000000000;
11'h333: mem_out = 27'b000000000000000000000000000;
11'h334: mem_out = 27'b000000000000000000000000000;
11'h335: mem_out = 27'b000000000000000000000000000;
11'h336: mem_out = 27'b000000000000000000000000000;
11'h337: mem_out = 27'b000000000000000000000000000;
11'h338: mem_out = 27'b000000000000000000000000000;
11'h339: mem_out = 27'b000000000000000000000000000;
11'h33A: mem_out = 27'b000000000000000000000000000;
11'h33B: mem_out = 27'b000000000000000000000000000;
11'h33C: mem_out = 27'b000000000000000000000000000;
11'h33D: mem_out = 27'b000000000000000000000000000;
11'h33E: mem_out = 27'b000000000000000000000000000;
11'h33F: mem_out = 27'b000000000000000000000000000;
11'h340: mem_out = 27'b000000000000000000000000000;
11'h341: mem_out = 27'b000000000000000000000000000;
11'h342: mem_out = 27'b000000000000000000000000000;
11'h343: mem_out = 27'b000000000000000000000000000;
```

```
11'h344: mem_out = 27'b000000000000000000000000000;
11'h345: mem_out = 27'b000000000000000000000000000;
11'h346: mem_out = 27'b000000000000000000000000000;
11'h347: mem_out = 27'b000000000000000000000000000;
11'h348: mem_out = 27'b000000000000000000000000000;
11'h349: mem_out = 27'b000000000000000000000000000;
11'h34A: mem_out = 27'b000000000000000000000000000;
11'h34B: mem_out = 27'b000000000000000000000000000;
11'h34C: mem_out = 27'b000000000000000000000000000;
11'h34D: mem_out = 27'b000000000000000000000000000;
11'h34E: mem_out = 27'b000000000000000000000000000;
11'h34F: mem_out = 27'b000000000000000000000000000;
11'h350: mem_out = 27'b000000000000000000000000000;
11'h351: mem_out = 27'b000000000000000000000000000;
11'h352: mem_out = 27'b000000000000000000000000000;
11'h353: mem_out = 27'b000000000000000000000000000;
11'h354: mem_out = 27'b000000000000000000000000000;
11'h355: mem_out = 27'b000000000000000000000000000;
11'h356: mem_out = 27'b000000000000000000000000000;
11'h357: mem_out = 27'b000000000000000000000000000;
11'h358: mem_out = 27'b000000000000000000000000000;
11'h359: mem_out = 27'b000000000000000000000000000;
11'h35A: mem_out = 27'b000000000000000000000000000;
11'h35B: mem_out = 27'b000000000000000000000000000;
11'h35C: mem_out = 27'b000000000000000000000000000;
11'h35D: mem_out = 27'b000000000000000000000000000;
11'h35E: mem_out = 27'b000000000000000000000000000;
11'h35F: mem_out = 27'b000000000000000000000000000;
11'h360: mem_out = 27'b000000000000000000000000000;
11'h361: mem_out = 27'b000000000000000000000000000;
11'h362: mem_out = 27'b000000000000000000000000000;
11'h363: mem_out = 27'b000000000000000000000000000;
11'h364: mem_out = 27'b000000000000000000000000000;
11'h365: mem_out = 27'b000000000000000000000000000;
11'h366: mem_out = 27'b000000000000000000000000000;
11'h367: mem_out = 27'b000000000000000000000000000;
11'h368: mem_out = 27'b000000000000000000000000000;
11'h369: mem_out = 27'b000000000000000000000000000;
11'h36A: mem_out = 27'b000000000000000000000000000;
11'h36B: mem_out = 27'b000000000000000000000000000;
11'h36C: mem_out = 27'b000000000000000000000000000;
11'h36D: mem_out = 27'b000000000000000000000000000;
11'h36E: mem_out = 27'b000000000000000000000000000;
11'h36F: mem_out = 27'b000000000000000000000000000;
```

```
11'h370: mem_out = 27'b000000000000000000000000000;
11'h371: mem_out = 27'b000000000000000000000000000;
11'h372: mem_out = 27'b000000000000000000000000000;
11'h373: mem_out = 27'b000000000000000000000000000;
11'h374: mem_out = 27'b000000000000000000000000000;
11'h375: mem_out = 27'b000000000000000000000000000;
11'h376: mem_out = 27'b000000000000000000000000000;
11'h377: mem_out = 27'b000000000000000000000000000;
11'h378: mem_out = 27'b000000000000000000000000000;
11'h379: mem_out = 27'b000000000000000000000000000;
11'h37A: mem_out = 27'b000000000000000000000000000;
11'h37B: mem_out = 27'b000000000000000000000000000;
11'h37C: mem_out = 27'b000000000000000000000000000;
11'h37D: mem_out = 27'b000000000000000000000000000;
11'h37E: mem_out = 27'b000000000000000000000000000;
11'h37F: mem_out = 27'b000000000000000000000000000;
11'h380: mem_out = 27'b000000000000000000000000000;
11'h381: mem_out = 27'b000000000000000000000000000;
11'h382: mem_out = 27'b000000000000000000000000000;
11'h383: mem_out = 27'b000000000000000000000000000;
11'h384: mem_out = 27'b000000000000000000000000000;
11'h385: mem_out = 27'b000000000000000000000000000;
11'h386: mem_out = 27'b000000000000000000000000000;
11'h387: mem_out = 27'b000000000000000000000000000;
11'h388: mem_out = 27'b000000000000000000000000000;
11'h389: mem_out = 27'b000000000000000000000000000;
11'h38A: mem_out = 27'b000000000000000000000000000;
11'h38B: mem_out = 27'b000000000000000000000000000;
11'h38C: mem_out = 27'b000000000000000000000000000;
11'h38D: mem_out = 27'b000000000000000000000000000;
11'h38E: mem_out = 27'b000000000000000000000000000;
11'h38F: mem_out = 27'b000000000000000000000000000;
11'h390: mem_out = 27'b000000000000000000000000000;
11'h391: mem_out = 27'b000000000000000000000000000;
11'h392: mem_out = 27'b000000000000000000000000000;
11'h393: mem_out = 27'b000000000000000000000000000;
11'h394: mem_out = 27'b000000000000000000000000000;
11'h395: mem_out = 27'b000000000000000000000000000;
11'h396: mem_out = 27'b000000000000000000000000000;
11'h397: mem_out = 27'b000000000000000000000000000;
11'h398: mem_out = 27'b000000000000000000000000000;
11'h399: mem_out = 27'b000000000000000000000000000;
11'h39A: mem_out = 27'b000000000000000000000000000;
11'h39B: mem_out = 27'b000000000000000000000000000;
```

```
11'h39C: mem_out = 27'b000000000000000000000000000;
11'h39D: mem_out = 27'b000000000000000000000000000;
11'h39E: mem_out = 27'b000000000000000000000000000;
11'h39F: mem_out = 27'b000000000000000000000000000;
11'h3A0: mem_out = 27'b000000000000000000000000000;
11'h3A1: mem_out = 27'b000000000000000000000000000;
11'h3A2: mem_out = 27'b000000000000000000000000000;
11'h3A3: mem_out = 27'b000000000000000000000000000;
11'h3A4: mem_out = 27'b000000000000000000000000000;
11'h3A5: mem_out = 27'b000000000000000000000000000;
11'h3A6: mem_out = 27'b000000000000000000000000000;
11'h3A7: mem_out = 27'b000000000000000000000000000;
11'h3A8: mem_out = 27'b000000000000000000000000000;
11'h3A9: mem_out = 27'b000000000000000000000000000;
11'h3AA: mem_out = 27'b000000000000000000000000000;
11'h3AB: mem_out = 27'b000000000000000000000000000;
11'h3AC: mem_out = 27'b000000000000000000000000000;
11'h3AD: mem_out = 27'b000000000000000000000000000;
11'h3AE: mem_out = 27'b000000000000000000000000000;
11'h3AF: mem_out = 27'b000000000000000000000000000;
11'h3B0: mem_out = 27'b000000000000000000000000000;
11'h3B1: mem_out = 27'b000000000000000000000000000;
11'h3B2: mem_out = 27'b000000000000000000000000000;
11'h3B3: mem_out = 27'b000000000000000000000000000;
11'h3B4: mem_out = 27'b000000000000000000000000000;
11'h3B5: mem_out = 27'b000000000000000000000000000;
11'h3B6: mem_out = 27'b000000000000000000000000000;
11'h3B7: mem_out = 27'b000000000000000000000000000;
11'h3B8: mem_out = 27'b000000000000000000000000000;
11'h3B9: mem_out = 27'b000000000000000000000000000;
11'h3BA: mem_out = 27'b000000000000000000000000000;
11'h3BB: mem_out = 27'b000000000000000000000000000;
11'h3BC: mem_out = 27'b000000000000000000000000000;
11'h3BD: mem_out = 27'b000000000000000000000000000;
11'h3BE: mem_out = 27'b000000000000000000000000000;
11'h3BF: mem_out = 27'b000000000000000000000000000;
11'h3C0: mem_out = 27'b000000000000000000000000000;
11'h3C1: mem_out = 27'b000000000000000000000000000;
11'h3C2: mem_out = 27'b000000000000000000000000000;
11'h3C3: mem_out = 27'b000000000000000000000000000;
11'h3C4: mem_out = 27'b000000000000000000000000000;
11'h3C5: mem_out = 27'b000000000000000000000000000;
11'h3C6: mem_out = 27'b000000000000000000000000000;
11'h3C7: mem_out = 27'b000000000000000000000000000;
```

```
11'h3C8: mem_out = 27'b000000000000000000000000000;
11'h3C9: mem_out = 27'b000000000000000000000000000;
11'h3CA: mem_out = 27'b000000000000000000000000000;
11'h3CB: mem_out = 27'b000000000000000000000000000;
11'h3CC: mem_out = 27'b000000000000000000000000000;
11'h3CD: mem_out = 27'b000000000000000000000000000;
11'h3CE: mem_out = 27'b000000000000000000000000000;
11'h3CF: mem_out = 27'b000000000000000000000000000;
11'h3D0: mem_out = 27'b000000000000000000000000000;
11'h3D1: mem_out = 27'b000000000000000000000000000;
11'h3D2: mem_out = 27'b000000000000000000000000000;
11'h3D3: mem_out = 27'b000000000000000000000000000;
11'h3D4: mem_out = 27'b000000000000000000000000000;
11'h3D5: mem_out = 27'b000000000000000000000000000;
11'h3D6: mem_out = 27'b000000000000000000000000000;
11'h3D7: mem_out = 27'b000000000000000000000000000;
11'h3D8: mem_out = 27'b000000000000000000000000000;
11'h3D9: mem_out = 27'b000000000000000000000000000;
11'h3DA: mem_out = 27'b000000000000000000000000000;
11'h3DB: mem_out = 27'b000000000000000000000000000;
11'h3DC: mem_out = 27'b000000000000000000000000000;
11'h3DD: mem_out = 27'b000000000000000000000000000;
11'h3DE: mem_out = 27'b000000000000000000000000000;
11'h3DF: mem_out = 27'b000000000000000000000000000;
11'h3E0: mem_out = 27'b000000000000000000000000000;
11'h3E1: mem_out = 27'b000000000000000000000000000;
11'h3E2: mem_out = 27'b000000000000000000000000000;
11'h3E3: mem_out = 27'b000000000000000000000000000;
11'h3E4: mem_out = 27'b000000000000000000000000000;
11'h3E5: mem_out = 27'b000000000000000000000000000;
11'h3E6: mem_out = 27'b000000000000000000000000000;
11'h3E7: mem_out = 27'b000000000000000000000000000;
11'h3E8: mem_out = 27'b000000000000000000000000000;
11'h3E9: mem_out = 27'b000000000000000000000000000;
11'h3EA: mem_out = 27'b000000000000000000000000000;
11'h3EB: mem_out = 27'b000000000000000000000000000;
11'h3EC: mem_out = 27'b000000000000000000000000000;
11'h3ED: mem_out = 27'b000000000000000000000000000;
11'h3EE: mem_out = 27'b000000000000000000000000000;
11'h3EF: mem_out = 27'b000000000000000000000000000;
11'h3F0: mem_out = 27'b000000000000000000000000000;
11'h3F1: mem_out = 27'b000000000000000000000000000;
11'h3F2: mem_out = 27'b000000000000000000000000000;
11'h3F3: mem_out = 27'b000000000000000000000000000;
```

```
                11'h3F4: mem_out = 27'b000000000000000000000000000;
                11'h3F5: mem_out = 27'b000000000000000000000000000;
                11'h3F6: mem_out = 27'b000000000000000000000000000;
                11'h3F7: mem_out = 27'b000000000000000000000000000;
                11'h3F8: mem_out = 27'b000000000000000000000000000;
                11'h3F9: mem_out = 27'b000000000000000000000000000;
                11'h3FA: mem_out = 27'b000000000000000000000000000;
                11'h3FB: mem_out = 27'b000000000000000000000000000;
                11'h3FC: mem_out = 27'b000000000000000000000000000;
                11'h3FD: mem_out = 27'b000000000000000000000000000;
                11'h3FE: mem_out = 27'b000000000000000000000000000;
                11'h3FF: mem_out = 27'b000000000000000000000000000;

            endcase

        end
endmodule
```