```
//*****************************************************************************************
//Hassen Abdu
//*****************************************************************************************
`timescale 1ns/1ns

//topVideo-instantiates topROM and myvga2

module topVideo(reset,clk,station,status,ebadbusy,RAMdata,vga_out_red,vga_out_green,vga_out_blue,
      vga_out_sync_b,vga_out_blank_b,vga_out_pixel_clock,vga_out_hsync,vga_out_vsync,RAMaddress,mebusy);



input reset,clk;

input station;
input[1:0] status;
input ebadbusy;       //busy signal from Ebad
input[15:0] RAMdata;  //datain from Ebad's RAM

output[7:0] vga_out_red, vga_out_green,vga_out_blue;
output vga_out_sync_b,vga_out_blank_b,vga_out_pixel_clock,vga_out_hsync,vga_out_vsync;

output mebusy;
output[9:0] RAMaddress;  //address to RAM

wire[3:0] horiz_count,vert_count;
wire[6:0] character;
wire[7:0] pixelrom;

wire vga_out_pixel_clock;
wire inv_vga_out_pixel_clock;
assign inv_vga_out_pixel_clock = ~vga_out_pixel_clock;  //pixel clock was inverted

wire[6:0] horiz_char,vert_char;

wire[6:0] romCharacter;
wire RAMselect;

vga3 myvga3(reset, clk, pixelrom, vga_out_red, vga_out_green, vga_out_blue,
         vga_out_sync_b, vga_out_blank_b, vga_out_pixel_clock,
         vga_out_hsync, vga_out_vsync,vert_count,horiz_count,vert_char,horiz_char,character);

topROM mytopROM(inv_vga_out_pixel_clock,character,ROMcharacter,RAMselect,station,status,horiz_char,
vert_char,horiz_count,
                  vert_count,pixelrom);

topRAM mytopRAM(inv_vga_out_pixel_clock,vert_char,horiz_char,ebadbusy,RAMdata,RAMselect,RAMaddress,
```

```
ROMcharacter,mebusy);

endmodule

//
//////////////////////////////////////////////////////////////////////////////////
`timescale 1ns/1ns
//ADClatch-latches through parallel signal from ADC conversion enabled on "ready" signal

module ADClatch(clk,ready,PDO,PDOlatched);

input clk,ready;
input[13:0] PDO;

output[13:0] PDOlatched;
reg[13:0] PDOlatched;

always @ (posedge clk) begin
if (ready) PDOlatched<=PDO;
else PDOlatched<=PDOlatched;
end

endmodule
//////////////////////////////////////////////////////////////////////////////////

`timescale 1ns/1ps
//FFTread-module that interprets the FFT module appropriately

module FFTread(pixel_clock,ramdata,vert_char,romcharacter);

input pixel_clock;
input[15:0] ramdata;
input[6:0] vert_char;
output[6:0] romcharacter;
reg[6:0] romcharacter;


parameter cutoff3=2;
parameter cutoff4=4;
parameter cutoff5=8;
parameter cutoff6=16;
parameter cutoff7=32;
parameter cutoff8=64;
parameter cutoff9=128;
parameter cutoff10=256;
parameter cutoff11=512;
parameter cutoff12=1024;
parameter cutoff13=2048;
```

```
parameter cutoff14=4096;
parameter cutoff15=8192;
parameter cutoff16=16384;
parameter cutoff17=32768;
parameter cutoff18=65536;


always @ (posedge pixel_clock) begin
	case (vert_char)
	31: if (ramdata>cutoff18) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	32: if (ramdata>cutoff17) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	33: if (ramdata>cutoff16) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	34: if (ramdata>cutoff15) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	35: if (ramdata>cutoff14) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	36: if (ramdata>cutoff13) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	37: if (ramdata>cutoff12) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	38: if (ramdata>cutoff11) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	39: if (ramdata>cutoff10) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	40: if (ramdata>cutoff9) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	41: if (ramdata>cutoff8) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	42: if (ramdata>cutoff7) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	43: if (ramdata>cutoff6) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	44: if (ramdata>cutoff5) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	45: if (ramdata>cutoff4) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	46: if (ramdata>cutoff3) romcharacter<=7'd42;
		else romcharacter<=7'd32;
	default: romcharacter<=7'd32;
	endcase
end
endmodule

//***************************************************************************
```

```
`timescale 1ns/1ns
//ADCserpar--serial to parallel conversion for the ADC

module ADCserpar(clk,conv,SDO,PDO,ready);

//clk is same as SCK from datasheet (~55Mhz)
//conv initiates ADC conversion
//SDO is from output of ADC in serial
//parallel is parallel output
//ready signal tells DSP that parallel signal is ready

input clk,conv;
input SDO;

output ready;
reg ready;

output[13:0] PDO;
reg[13:0] PDO;


reg par0,par1,par2,par3,par4,par5,par6,par7,par8,par9,par10,par11,
par12,par13,par14,par15,par16,par17;
//18 registers because 18-clkcycle frame

assign invclk=~clk;

always @ (posedge invclk) begin     //negedge of clk
if (!conv) ready<=1;          //conv is active low
else ready<=0;

par17<=par16;
par16<=par15;
par15<=par14;
par14<=par13;
par13<=par12;
par12<=par11;
par11<=par10;
par10<=par9;
par9<=par8;
par8<=par7;
par7<=par6;
par6<=par5;
par5<=par4;
par4<=par3;
par3<=par2;
par2<=par1;
par1<=par0;
```

```
par0<=SDO;

PDO<={par15,par14,par13,par12,par11,par10,par9,par8,par7,par6,par5,par4,par3,par2};
end
endmodule
//**********************************************************************************
`timescale 1ns/1ns

//RAMcontrol-module that will store the user selctions made


module RAMcontrol(pixel_clk,horiz_char,vert_char,ebadbusy,
                  RAMselect,RAMaddress,mebusy);

input pixel_clk;
input[6:0] horiz_char,vert_char;
input ebadbusy;   //tells me that Ebad is busy

output RAMselect;
reg RAMselect;
output[9:0] RAMaddress;
reg[9:0] RAMaddress;
output mebusy;   //to tell Ebad I'm busy reading
reg mebusy;

parameter factor=3;
parameter offset=200;

always @ (posedge pixel_clk) begin
      if (ebadbusy) begin
            RAMselect<=0;
            RAMaddress<=10'd0;
            mebusy<=0;
            end
      else if ((vert_char>30) && (vert_char<49) && (horiz_char>3) && (horiz_char<95)) begin
            mebusy<=1;
            RAMselect<=1;
            RAMaddress<=((factor*horiz_char)+offset);
            end
      else begin
            mebusy<=0;
            RAMselect<=0;
            RAMaddress<=10'd0;
            end
end
endmodule
//**********************************************************************************
`timescale 1ns/1ns
```

```
//RAMcontrol-module that will store the user selctions made


module RAMcontrol(pixel_clk,horiz_char,vert_char,ebadbusy,
                  RAMselect,RAMaddress,mebusy);

input pixel_clk;
input[6:0] horiz_char,vert_char;
input ebadbusy;   //tells me that Ebad is busy

output RAMselect;
reg RAMselect;
output[9:0] RAMaddress;
reg[9:0] RAMaddress;
output mebusy;   //to tell Ebad I'm busy reading
reg mebusy;

parameter factor=3;
parameter offset=200;

always @ (posedge pixel_clk) begin
      if (ebadbusy) begin
            RAMselect<=0;
            RAMaddress<=10'd0;
            mebusy<=0;
            end
      else if ((vert_char>30) && (vert_char<49) && (horiz_char>3) && (horiz_char<95)) begin
            mebusy<=1;
            RAMselect<=1;
            RAMaddress<=((factor*horiz_char)+offset);
            end
      else begin
            mebusy<=0;
//RAMcontrol-module that will store the user selctions made


module RAMcontrol(pixel_clk,horiz_char,vert_char,ebadbusy,
                  RAMselect,RAMaddress,mebusy);

input pixel_clk;
input[6:0] horiz_char,vert_char;
input ebadbusy;   //tells me that Ebad is busy

output RAMselect;
reg RAMselect;
output[9:0] RAMaddress;
reg[9:0] RAMaddress;
```

```
output mebusy;   //to tell Ebad I'm busy reading
reg mebusy;

parameter factor=3;
parameter offset=200;

always @ (posedge pixel_clk) begin
      if (ebadbusy) begin
            RAMselect<=0;
            RAMaddress<=10'd0;
            mebusy<=0;
            end
      else if ((vert_char>30) && (vert_char<49) && (horiz_char>3) && (horiz_char<95)) begin
            mebusy<=1;
            RAMselect<=1;
            RAMaddress<=((factor*horiz_char)+offset);
            end
      else begin
            mebusy<=0;
            RAMselect<=0;
            RAMaddress<=10'd0;
            end
end
endmodule
//**********************************************************************
/**********************************************************************
*    This file is owned and controlled by Xilinx and must be used      *
*    solely for design, simulation, implementation and creation of     *
*    design files limited to Xilinx devices or technologies. Use       *
*    with non-Xilinx devices or technologies is expressly prohibited   *
*    and immediately terminates your license.                          *
*                                                                      *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"     *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR           *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION   *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION       *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS         *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,           *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE  *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY          *
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE           *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR    *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF   *
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS   *
*    FOR A PARTICULAR PURPOSE.                                         *
*                                                                      *
*    Xilinx products are not intended for use in life support          *
*    appliances, devices, or systems. Use in such applications are     *
```

```
*    expressly prohibited.                                   *
*                                                            *
*    (c) Copyright 1995-2004 Xilinx, Inc.                    *
*    All rights reserved.                                    *
**************************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file characterrom.v when simulating
// the core, characterrom. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Guide".

module characterrom (
      addr,
      clk,
      dout);    // synthesis black_box

input [10 : 0] addr;
input clk;
output [7 : 0] dout;

// synopsys translate_off

      BLKMEMSP_V6_1 #(
            11,     // c_addr_width
            "0",    // c_default_data
            2048,   // c_depth
            0,      // c_enable_rlocs
            0,      // c_has_default_data
            0,      // c_has_din
            0,      // c_has_en
            0,      // c_has_limit_data_pitch
            0,      // c_has_nd
            0,      // c_has_rdy
            0,      // c_has_rfd
            0,      // c_has_sinit
            0,      // c_has_we
            18,     // c_limit_data_pitch
            "characterrom.mif",     // c_mem_init_file
            0,      // c_pipe_stages
            0,      // c_reg_inputs
            "0",    // c_sinit_value
            8,      // c_width
            0,      // c_write_mode
            "0",    // c_ybottom_addr
            1,      // c_yclk_is_rising
```

```
            1,      // c_yen_is_high
            "hierarchy1",   // c_yhierarchy
            0,      // c_ymake_bmm
            "16kx1",        // c_yprimitive_type
            1,      // c_ysinit_is_high
            "1024", // c_ytop_addr
            1,      // c_yuse_single_primitive
            1,      // c_ywe_is_high
            1)      // c_yydisable_warnings
        inst (
            .ADDR(addr),
            .CLK(clk),
            .DOUT(dout),
            .DIN(),
            .EN(),
            .ND(),
            .RFD(),
            .RDY(),
            .SINIT(),
            .WE());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of characterrom is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of characterrom is "black_box"

endmodule

//***************************************************************
`timescale 1ns/1ns

//ROM minor FSM---minor FSM that accesses ROM

module romFSM(clk,character,horiz_char,vert_char,ROMcharacter,RAMselect,horiz_count,vert_count,
            station,status,addressROM);

input clk;

input[6:0] character;

input[3:0] horiz_count, vert_count;
input [6:0] horiz_char,vert_char;
```

```
input station;   //0 -- 850, 1 -- 1030
input[1:0] status;  //00 - Playing, 01 - Paused, 10 - Replaying, 11 - Programmed
input[6:0] ROMcharacter;   //toROM-character value from RAM
input RAMselect;

output[10:0] addressROM;
reg[10:0] addressROM;

reg[10:0] addr2,addr1,addr;



always @ (posedge clk) begin
	addr1<=addr;
	addr2<=addr1;
	addressROM<=addr2;
	if (vert_char==13) begin
		if (station)
			case (horiz_char)
			19: addr<=12*55+vert_count;  //W
			20: addr<=12*34+vert_count;  //B
			21: addr<=12*58+vert_count;  //Z
			default: addr1<=12*(character-32)+vert_count;
			endcase
		else case (horiz_char)
			19: addr<=12*55+vert_count;  //W
			20: addr<=12*37+vert_count;  //E
			21: addr<=12*37+vert_count;  //E
			22: addr<=12*41+vert_count;  //I
			default: addr1<=12*(character-32)+vert_count;
			endcase
		end
	else if (vert_char==15) begin
		if (station)
			case (horiz_char)
				19: addr<=12*17+vert_count;  //1
				20: addr<=12*16+vert_count;  //0
				21: addr<=12*19+vert_count;  //3
				22: addr<=12*16+vert_count;  //0
				23: addr<=12*75+vert_count;  //k
			24: addr<=12*40+vert_count;  //H
			25: addr<=12*90+vert_count;  //z
				default: addr1<=12*(character-32)+vert_count;
				endcase
		else case (horiz_char)
			19: addr<=12*24+vert_count;  //8
			20: addr<=12*21+vert_count;  //5
			21: addr<=12*16+vert_count;  //0
```

```
                23: addr<=12*75+vert_count;  //k
            24: addr<=12*40+vert_count;  //H
            25: addr<=12*90+vert_count;  //z
                default: addr1<=12*(character-32)+vert_count;
                endcase
             end
        else if (vert_char==22) begin
            if (status==2'b01)
                case (horiz_char)
                16: addr<=12*50+vert_count;  //R
                17: addr<=12*69+vert_count;  //e
                18: addr<=12*80+vert_count;  //p
                19: addr<=12*76+vert_count;  //l
                20: addr<=12*65+vert_count;  //a
                21: addr<=12*89+vert_count;  //y
                22: addr<=12*73+vert_count;  //i
                23: addr<=12*78+vert_count;  //n
                24: addr<=12*71+vert_count;  //g
                default: addr1<=12*(character-32)+vert_count;
                endcase
            else if (status==2'b10)
                case (horiz_char)
                17: addr<=12*48+vert_count;  //P
                18: addr<=12*65+vert_count;  //a
                19: addr<=12*85+vert_count;  //u
                20: addr<=12*83+vert_count;  //s
                21: addr<=12*69+vert_count;  //e
                22: addr<=12*68+vert_count;  //d
                default: addr1<=12*(character-32)+vert_count;
                endcase
            else if (status==2'b11)
                case (horiz_char)
                15: addr<=12*48+vert_count;  //P
                16: addr<=12*82+vert_count;  //r
                17: addr<=12*79+vert_count;  //o
                18: addr<=12*71+vert_count;  //g
                19: addr<=12*82+vert_count;  //r
                20: addr<=12*65+vert_count;  //a
                21: addr<=12*77+vert_count;  //m
                22: addr<=12*77+vert_count;  //m
                23: addr<=12*69+vert_count;  //e
                24: addr<=12*68+vert_count;  //d
                default: addr1<=12*(character-32)+vert_count;
                endcase
            else case (horiz_char)
                17: addr<=12*48+vert_count;  //P
                18: addr<=12*76+vert_count;  //l
                19: addr<=12*65+vert_count;  //a
```

```
                20: addr<=12*89+vert_count;  //y
                21: addr<=12*73+vert_count;  //i
                22: addr<=12*78+vert_count;  //n
                23: addr<=12*71+vert_count;  //g
                default: addr1<=12*(character-32)+vert_count;
                endcase
            end
        else if (RAMselect) begin
            addr<=12*(ROMcharacter-32)+vert_count;
            end

        else begin
            addr1<=12*(character-32)+vert_count;
        end
        end

endmodule
//*****************************************************************
`timescale 1ns/1ns

module topADC(clk,Fsout,SDO,PDOlatched,ready);

input clk,Fsout;
input SDO;

output[13:0] PDOlatched;
output ready;

wire[13:0] PDO;

ADCserpar myADCserpar(clk,Fsout,SDO,PDO,ready);
ADClatch myADClatch(clk,ready,PDO,PDOlatched);

endmodule
//*********************************************************
//topRAM-top module for RAMs

module topRAM(pixel_clock,vert_char,horiz_char,ebadbusy,ramdata,RAMselect,RAMaddress,romCharacter,
mebusy);

input pixel_clock;
input[6:0] vert_char,horiz_char;
input ebadbusy;
input[15:0] ramdata;

output[6:0] romCharacter;
output mebusy,RAMselect;
```

```
output[9:0] RAMaddress;

RAMcontrol myRAMcontrol(pixel_clock,horiz_char,vert_char,ebadbusy,
                RAMselect,RAMaddress,mebusy);

FFTread myFFTread(pixel_clock,ramdata,vert_char,romCharacter);

endmodule
//*****************************************************************
`timescale 1ns/1ns

//topROM-access ROM to get character bitmaps

module topROM(clk,character,ROMcharacter,RAMselect,station,status,horiz_char,vert_char,
                horiz_count,vert_count,pixelrom);

input clk;
input RAMselect;
input[6:0] character;
input[6:0] ROMcharacter;  //ROMcharacter-from RAM
input[3:0] horiz_count, vert_count;
input[6:0] horiz_char,vert_char;
input[1:0] status;
input station;

output[7:0] pixelrom;

wire [10:0] addr;

romFSM myromFSM(clk,character,horiz_char,vert_char,ROMcharacter,RAMselect,horiz_count,vert_count,
            station,status,addr);

characterrom mycharacterrom(addr,clk,pixelrom);

endmodule

//*****************************************************************
`timescale 1ns/1ns

//topVideo-instantiates topROM and myvga2

module topVideo(reset,clk,station,status,ebadbusy,RAMdata,vga_out_red,vga_out_green,vga_out_blue,
      vga_out_sync_b,vga_out_blank_b,vga_out_pixel_clock,vga_out_hsync,vga_out_vsync,RAMaddress,mebusy);



input reset,clk;
```

```
input station;
input[1:0] status;
input ebadbusy;       //busy signal from Ebad
input[15:0] RAMdata;  //datain from Ebad's RAM

output[7:0] vga_out_red, vga_out_green,vga_out_blue;
output vga_out_sync_b,vga_out_blank_b,vga_out_pixel_clock,vga_out_hsync,vga_out_vsync;

output mebusy;
output[9:0] RAMaddress;  //address to RAM

wire[3:0] horiz_count,vert_count;
wire[6:0] character;
wire[7:0] pixelrom;

wire vga_out_pixel_clock;
wire inv_vga_out_pixel_clock;
assign inv_vga_out_pixel_clock = ~vga_out_pixel_clock;  //pixel clock was inverted

wire[6:0] horiz_char,vert_char;

wire[6:0] romCharacter;
wire RAMselect;

vga3 myvga3(reset, clk, pixelrom, vga_out_red, vga_out_green, vga_out_blue,
        vga_out_sync_b, vga_out_blank_b, vga_out_pixel_clock,
        vga_out_hsync, vga_out_vsync,vert_count,horiz_count,vert_char,horiz_char,character);

topROM mytopROM(inv_vga_out_pixel_clock,character,ROMcharacter,RAMselect,station,status,horiz_char,
vert_char,horiz_count,
                vert_count,pixelrom);

topRAM mytopRAM(inv_vga_out_pixel_clock,vert_char,horiz_char,ebadbusy,RAMdata,RAMselect,RAMaddress,
ROMcharacter,mebusy);

endmodule
//////////////////////////////////////////////////////////////////////////////////
`timescale 1ns/1ns

module vga3 (reset, clock_27mhz, pixelrom, vga_out_red, vga_out_green, vga_out_blue,
        vga_out_sync_b, vga_out_blank_b, vga_out_pixel_clock,
        vga_out_hsync, vga_out_vsync,vert_count,horiz_count,vert_char,horiz_char,character);

  input reset; // Active high reset, synchronous with 27MHz clock
  input clock_27mhz; // 27MHz input clock
  input[7:0] pixelrom;
  output [7:0] vga_out_red, vga_out_green, vga_out_blue; // Outputs to DAC
  output vga_out_sync_b, vga_out_blank_b; // Composite sync/blank outputs to DAC
```

```
  output vga_out_pixel_clock; // Pixel clock for DAC
  output vga_out_hsync, vga_out_vsync; // Sync outputs to VGA connector
  output[3:0] vert_count,horiz_count;
  output[6:0] character;
  output[6:0] vert_char,horiz_char;

  ////////////////////////////////////////////////////////////////////////
  //
  // Timing values
  //
  ////////////////////////////////////////////////////////////////////////

  // 800 X 600 @ 72Hz with a 50MHz pixel clock
`define H_ACTIVE            800 // pixels
`define H_FRONT_PORCH     56 // pixels
`define H_SYNCH          120 // pixels
`define H_BACK_PORCH      64 // pixels
`define H_TOTAL         1040 // pixels

`define V_ACTIVE            600 // lines
`define V_FRONT_PORCH     37 // lines
`define V_SYNCH            6 // lines
`define V_BACK_PORCH      23 // lines
`define V_TOTAL          666 // lines

`define H_CHARAC            8 //8 pixels wide characters
`define V_CHARAC           12 //12 pixels long characters

  ////////////////////////////////////////////////////////////////////////
  //
  // Internal signals
  //
  ////////////////////////////////////////////////////////////////////////

  wire pixel_clock;
  reg prst, pixel_reset; // Active high reset, synchronous with pixel clock

  reg [7:0] vga_out_red, vga_out_blue, vga_out_green;
  wire vga_out_sync_b, vga_out_blank_b;
  reg hsync1, hsync2, vga_out_hsync, vsync1, vsync2, vga_out_vsync;


  reg [10:0] pixel_count; // Counts pixels in each line
  reg [10:0] line_count; // Counts lines in each frame

  reg[3:0] vert_count,horiz_count;
  reg[6:0] vert_char,horiz_char;
  reg[6:0] character;
```

```
  reg pixel;
  reg[3:0] newhoriz_count,newhoriz_count2,newhoriz_count3,newhoriz_count4;
  reg[3:0] newvert_count,newvert_count2,newvert_count3;
  reg vert_off,horiz_off;
  wire inv_vga_out_hsync;
  reg vert_reset,vert_reset2;

  //////////////////////////////////////////////////////////////////////
  //
  // Generate the pixel clock (50MHz)
  //
  //////////////////////////////////////////////////////////////////////

  // synthesis attribute period of clock_27mhz is 37ns;
  assign pixel_clock=clock_27mhz;
//          DCM vga_dcm (.CLKIN(clock_27mhz),
//          .RST(1'b0),
//          .CLKFX(pixel_clock));
  // synthesis attribute DLL_FREQUENCY_MODE of vga_dcm is "LOW"
  // synthesis attribute DUTY_CYCLE_CORRECTION of vga_dcm is "TRUE"
  // synthesis attribute STARTUP_WAIT of vga_dcm is "TRUE"
  // synthesis attribute DFS_FREQUENCY_MODE of vga_dcm is "LOW"
  // synthesis attribute CLKFX_DIVIDE of vga_dcm is 7
  // synthesis attribute CLKFX_MULTIPLY of vga_dcm is 13
  // synthesis attribute CLK_FEEDBACK of vga_dcm  is "1X"
  // synthesis attribute CLKOUT_PHASE_SHIFT of vga_dcm is "NONE"
  // synthesis attribute PHASE_SHIFT of vga_dcm is 0
  // synthesis attribute clkin_period of vga_dcm is "37.04ns"

  assign vga_out_pixel_clock = ~pixel_clock;

  always @(posedge pixel_clock)
    begin
      prst <= reset;
      pixel_reset <= prst;
      vert_reset <= pixel_reset;
      vert_reset2 <=vert_reset;
    end

  //////////////////////////////////////////////////////////////////////
  //
  // Pixel and Line Counters
  //
  //////////////////////////////////////////////////////////////////////

  always @(posedge pixel_clock)
    if (pixel_reset)
      begin
```

```
      pixel_count <= 0;
      line_count <= 0;
      vert_count<=0;
      vert_char<=0;
      end
  else if (pixel_count == (`H_TOTAL-1)) // last pixel in the line
    begin
      pixel_count <= 0;
      if (line_count == (`V_TOTAL-1)) // last line of the frame
        begin
        line_count<=0;
        vert_char<=0;
        vert_count<=0;
        end
      else begin
          if (vert_count==(`V_CHARAC-1)) begin
                vert_char<=vert_char+1;
                vert_count<=0;
                line_count <= line_count + 1;
                end
          else begin
                line_count<=line_count+1;
                vert_count<=vert_count+1;
                vert_char<=vert_char;
                end
          end
       end
  else      begin
    pixel_count <= pixel_count +1;
     end

always @ (posedge pixel_clock) begin
          newhoriz_count<=horiz_count;        //to keep the horiz_count for pixel
          newhoriz_count2<=newhoriz_count;
          newhoriz_count3<=newhoriz_count2;
          newhoriz_count4<=newhoriz_count3;
          if (pixel_reset)
          begin
                horiz_char  <= 0;   //# of character horizontally
                horiz_count <= (`H_CHARAC-1);   //horizontal counter
                horiz_off<=0;
                end
     else begin
          if (!vga_out_hsync) begin
                      horiz_char<=0;
                      horiz_count<=(`H_CHARAC-1);
                      end
                else if (!vga_out_blank_b) begin
```

```
            horiz_char<=horiz_char;
              horiz_count<=horiz_count;
                    end
              else if (horiz_count == 0) begin  //8th character pixel
            horiz_char<=horiz_char+1;
              horiz_count<=(`H_CHARAC-1);
        end
                    else begin
                          horiz_char<=horiz_char;
              horiz_count<=horiz_count-1;
              end
              end
        end

//////////////////////////////////////////////////////////////////////
//
// Sync and Blank Signals
//
//////////////////////////////////////////////////////////////////////

always @ (posedge pixel_clock)
  begin
    if (pixel_reset)
      begin
        hsync1 <= 1;
        hsync2 <= 1;
        vga_out_hsync <= 1;
        vsync1 <= 1;
        vsync2 <= 1;
        vga_out_vsync <= 1;
      end
    else
      begin
           // Horizontal sync
        if (pixel_count == (`H_ACTIVE+`H_FRONT_PORCH))
          hsync1 <= 0; // start of h_sync
        else if (pixel_count == (`H_ACTIVE+`H_FRONT_PORCH+`H_SYNCH))
          hsync1 <= 1; // end of h_sync

        // Vertical sync
        if (pixel_count == (`H_TOTAL-1))
          begin
            if (line_count == (`V_ACTIVE+`V_FRONT_PORCH))
              vsync1 <= 0; // start of v_sync
            else if (line_count == (`V_ACTIVE+`V_FRONT_PORCH+`V_SYNCH))
              vsync1 <= 1; // end of v_sync
          end
        end
```

```
    // Delay hsync and vsync by two cycles to compensate for 2 cycles of
    // pipeline delay in the DAC.
    hsync2 <= hsync1;
    vga_out_hsync <= hsync2;

    vsync2 <= vsync1;
    vga_out_vsync <= vsync2;

  end

// Blanking
assign vga_out_blank_b = ((pixel_count<`H_ACTIVE) & (line_count<`V_ACTIVE));

// Composite sync
assign vga_out_sync_b = hsync1 ^ vsync1;

/////inverter Hsync
assign inv_vga_out_hsync=~vga_out_hsync;


//////////////////////////////////////////////////////////////////////
//
// Generate a pretty picture
//
//////////////////////////////////////////////////////////////////////




always @ (posedge pixel_clock) begin
      //maximum of 80 characters wide, 32 characters tall
    case (vert_char)
      0: case (horiz_char)              //outputs "SPRING 2005" completely
        30:character<=7'd77; //M
            31:character<=7'd97; //a
            32:character<=7'd115;       //s
            33:character<=7'd115;  //s
        34:character<=7'd97;  //a
        35:character<=7'd99;  //c
        36:character<=7'd104;  //h
        37:character<=7'd117;  //u
        38:character<=7'd115;  //s
        39:character<=7'd101;   //e
        40:character<=7'd116;   //t
        41:character<=7'd116;   //t
        42:character<=7'd115;   //s
```

```
44:character<=7'd73;  // I
45:character<=7'd110;  //n
    46:character<=7'd115;       //s
    47:character<=7'd116;       //t
    48:character<=7'd105;       //i
    49:character<=7'd116;   //t
  50:character<=7'd117;  //u
51:character<=7'd116;  //t
52:character<=7'd101;  //e
54:character<=7'd111;  //o
55:character<=7'd102;  //f
57:character<=7'd84;   //T
58:character<=7'd101;   //e
59:character<=7'd99;   //c
60:character<=7'd104;   //h
61:character<=7'd110;  //n
62:character<=7'd111;  //o
    63:character<=7'd108;       //l
    64:character<=7'd111;       //o
    65:character<=7'd103;       //g
    66:character<=7'd121;  //y
default: character<=7'd32;  //(space)
 endcase

   1: case (horiz_char)
26:character<=7'd54;        //6
    27:character<=7'd46;   //.
    28:character<=7'd49;   //1
    29:character<=7'd49;    //1
30:character<=7'd49;    //1
32:character<=7'd73;    //I
33:character<=7'd110;   //n
34:character<=7'd116;   //t
35:character<=7'd114;   //r
36:character<=7'd111;   //o
37:character<=7'd100;   //d
38:character<=7'd117;   //u
39:character<=7'd99;    //c
40:character<=7'd116;   //t
41:character<=7'd111;   //o
    42:character<=7'd114;         //r
    43:character<=7'd121;         //y
    45:character<=7'd68;   //D
    46:character<=7'd105;   //i
47:character<=7'd103;   //g
48:character<=7'd105;   //i
49:character<=7'd116;   //t
50:character<=7'd97;    //a
```

```
    51:character<=7'd108;   //l
    54:character<=7'd83;    //S
    55:character<=7'd121;   //y
    56:character<=7'd115;   //s
    57:character<=7'd116;   //t
    58:character<=7'd101;   //e
    59:character<=7'd109;   //m
        60:character<=7'd115;         //s
        62:character<=7'd76;   //L
        63:character<=7'd97;   //a
        64:character<=7'd98;    //b
    65:character<=7'd111;   //o
    66:character<=7'd114;   //r
    67:character<=7'd97;    //a
    68:character<=7'd116;   //t
    69:character<=7'd111;   //o
    70:character<=7'd114;   //r
        71:character<=7'd121;   //y
        default: character<=7'd32;
        endcase

  2: case (horiz_char)
        43:character<=7'd83;   //S
    44:character<=7'd112;  //p
    45:character<=7'd114;  //r
    46:character<=7'd105;  //i
    47:character<=7'd110;  //n
    48:character<=7'd103;  //g
    50:character<=7'd50;   //2
    51:character<=7'd48;   //0
    52:character<=7'd48;   //0
    53:character<=7'd53;  //5
    default: character<=7'd32;  //(space)
     endcase

   4: case (horiz_char)
        47:character<=7'd66;   //B
    48:character<=7'd89;   //Y
    default: character<=7'd32;  //(space)
     endcase

      5: case (horiz_char)
        42:character<=7'd72;   //H
    43:character<=7'd97;   //a
    44:character<=7'd115;  //s
    45:character<=7'd115;  //s
    46:character<=7'd101;  //e
    47:character<=7'd110;  //n
```

```
49:character<=7'd65;   //A
50:character<=7'd98;   //b
51:character<=7'd100;  //d
52:character<=7'd117;  //u
default: character<=7'd32;  //(space)
 endcase

   6: case (horiz_char)
      42:character<=7'd69;   //E
43:character<=7'd98;   //b
44:character<=7'd97;   //a
45:character<=7'd100;  //d
47:character<=7'd65;   //A
48:character<=7'd104;  //h
49:character<=7'd109;  //m
50:character<=7'd101;  //e
51:character<=7'd100;  //d
default: character<=7'd32;  //(space)
 endcase

    7: case (horiz_char)
      42:character<=7'd87;   //W
43:character<=7'd97;   //a
44:character<=7'd106;  //j
45:character<=7'd97;   //a
46:character<=7'd104;  //h
47:character<=7'd97;   //a
48:character<=7'd116;  //t
50:character<=7'd75;   //K
51:character<=7'd104;  //h
52:character<=7'd97;   //a
      53:character<=7'd110;  //n
default: character<=7'd32;  //(space)
 endcase

   11: case (horiz_char)
      12:character<=7'd67;    //C
13:character<=7'd104;   //h
14:character<=7'd97;    //a
15:character<=7'd110;   //n
16:character<=7'd110;   //n
17:character<=7'd101;   //e
18:character<=7'd108;   //l
20:character<=7'd73;    //I
21:character<=7'd110;   //n
      22:character<=7'd102;   //f
      23:character<=7'd111;   //o
      24:character<=7'd114;   //r
```

```
25:character<=7'd109;   //m
26:character<=7'd97;    //a
    27:character<=7'd116;   //t
28:character<=7'd105;   //i
29:character<=7'd111;   //o
30:character<=7'd110;   //n
62:character<=7'd85;    //U
63:character<=7'd115;   //s
64:character<=7'd101;   //e
    65:character<=7'd114;   //r
    67:character<=7'd77;    //M
68:character<=7'd101;   //e
69:character<=7'd110;   //n
70:character<=7'd117;   //u
default: character<=7'd32;  //(space)
 endcase

   13: case (horiz_char)
    12:character<=7'd78;   //N
13:character<=7'd97;   //a
14:character<=7'd109;  //m
15:character<=7'd101;  //e
16:character<=7'd58;   //:
    52:character<=7'd83;   //S
53:character<=7'd119;  //w
54:character<=7'd105;  //i
    55:character<=7'd116;  //t
56:character<=7'd99;   //c
57:character<=7'd104;  //h
59:character<=7'd35;   //#
60:character<=7'd48;   //0
61:character<=7'd58;   //:
63:character<=7'd83;   //S
    64:character<=7'd101;  //e
    65:character<=7'd108;  //l
66:character<=7'd101;  //e
67:character<=7'd99;   //c
68:character<=7'd116;  //t
    70:character<=7'd82;   //R
71:character<=7'd97;   //a
    72:character<=7'd100;  //d
73:character<=7'd105;  //i
    74:character<=7'd111;  //o
76:character<=7'd67;   //C
    77:character<=7'd104;  //h
78:character<=7'd97;   //a
    79:character<=7'd110;  //n
80:character<=7'd110;  //n
```

```
      81:character<=7'd101;  //e
    82:character<=7'd108;  //l
      /*19:character<=7'd101; //RAM
    20:character<=7'd101;  //RAM
      21:character<=7'd101;  //RAM
    22:character<=7'd101;  //RAM  */
      default: character<=7'd32;  //(space)
   endcase

     15: case (horiz_char)
      10: character<=7'd78;   //N
    11:character<=7'd117;   //u
      12:character<=7'd109;   //m
    13:character<=7'd98;    //b
      14:character<=7'd101;   //e
    15:character<=7'd114;   //r
    16:character<=7'd58;    //:
    52:character<=7'd83;    //S
      53:character<=7'd119;   //w
      54:character<=7'd105;   //i
    55:character<=7'd116;   //t
    56:character<=7'd99;    //c
    57:character<=7'd104;   //h
      59:character<=7'd35;    //#
    60:character<=7'd49;    //1
      61:character<=7'd58;    //:
    63:character<=7'd80;    //P
      64:character<=7'd97;    //a
    65:character<=7'd117;   //u
      66:character<=7'd115;   //s
    67:character<=7'd101;   //e
      69:character<=7'd76;    //L
    70:character<=7'd105;   //i
      71:character<=7'd118;   //v
    72:character<=7'd101;   //e
      74:character<=7'd66;    //B
      75:character<=7'd114;   //r
      76:character<=7'd111;   //o
    77:character<=7'd97;    //a
    78:character<=7'd100;   //d
    79:character<=7'd99;    //c
      80:character<=7'd97;    //a
    81:character<=7'd115;   //s
      82:character<=7'd116;   //t
    /*19:character<=7'd101; //RAM
    20:character<=7'd101;   //RAM
    21:character<=7'd101;   //RAM
    22:character<=7'd101;   //RAM */
```

```
        default: character<=7'd32;  //(space)
    endcase

      17: case (horiz_char)
        52:character<=7'd83;    //S
        53:character<=7'd119;   //w
        54:character<=7'd105;   //i
    55:character<=7'd116;   //t
    56:character<=7'd99;    //c
    57:character<=7'd104;   //h
        59:character<=7'd35;    //#
    60:character<=7'd50;    //2
        61:character<=7'd58;    //:
        63:character<=7'd82;    //R
    64:character<=7'd101;   //e
        65:character<=7'd112;   //p
        66:character<=7'd108;   //l
    67:character<=7'd97;    //a
        68:character<=7'd121;   //y
    70:character<=7'd82;    //R
    71:character<=7'd101;   //e
        72:character<=7'd99;    //c
        73:character<=7'd111;   //o
        74:character<=7'd114;   //r
    75:character<=7'd100;   //d
    76:character<=7'd101;   //e
    77:character<=7'd100;   //d
        79:character<=7'd66;    //B
        80:character<=7'd114;   //r
        81:character<=7'd111;   //o
    82:character<=7'd97;    //a
    83:character<=7'd100;   //d
    84:character<=7'd99;    //c
        85:character<=7'd97;    //a
    86:character<=7'd115;   //s
        87:character<=7'd116;   //t
    default: character<=7'd32;  //(space)
  endcase

      19: case (horiz_char)
        52:character<=7'd83;    //S
        53:character<=7'd119;   //w
        54:character<=7'd105;   //i
    55:character<=7'd116;   //t
    56:character<=7'd99;    //c
    57:character<=7'd104;   //h
        59:character<=7'd35;    //#
    60:character<=7'd51;    //3
```

```
        61:character<=7'd58;    //:
        63:character<=7'd80;    //P
    64:character<=7'd114;   //r
        65:character<=7'd111;   //o
        66:character<=7'd103;   //g
    67:character<=7'd114;   //r
        68:character<=7'd97;    //a
    69:character<=7'd109;   //m
    71:character<=7'd80;    //P
        72:character<=7'd114;   //r
        73:character<=7'd101;   //e
        74:character<=7'd115;   //s
    75:character<=7'd101;   //e
    76:character<=7'd116;   //t
    78:character<=7'd49;    //1
        default: character<=7'd32;  //(space)
  endcase

      20: case (horiz_char)
        14:character<=7'd83;    //S
        15:character<=7'd121;   //y
        16:character<=7'd115;   //s
    17:character<=7'd116;   //t
    18:character<=7'd101;   //e
    19:character<=7'd109;   //m
        21:character<=7'd83;    //S
    22:character<=7'd116;   //t
        23:character<=7'd97;    //a
        24:character<=7'd116;   //t
    25:character<=7'd117;   //u
        26:character<=7'd115;   //s
        default: character<=7'd32;  //(space)
    endcase

      21: case (horiz_char)
        52:character<=7'd83;    //S
        53:character<=7'd119;   //w
        54:character<=7'd105;   //i
    55:character<=7'd116;   //t
    56:character<=7'd99;    //c
    57:character<=7'd104;   //h
        59:character<=7'd35;    //#
    60:character<=7'd52;    //4
        61:character<=7'd58;    //:
        63:character<=7'd80;    //P
    64:character<=7'd114;   //r
        65:character<=7'd111;   //o
        66:character<=7'd103;   //g
```

```
67:character<=7'd114;   //r
    68:character<=7'd97;    //a
69:character<=7'd109;   //m
71:character<=7'd80;    //P
    72:character<=7'd114;   //r
    73:character<=7'd101;   //e
    74:character<=7'd115;   //s
75:character<=7'd101;   //e
76:character<=7'd116;   //t
78:character<=7'd50;    //2
    default: character<=7'd32; //(space)
    endcase

  /*22: case (horiz_char)
    15:character<=7'd101;    //RAM
16:character<=7'd101;    //RAM
17:character<=7'd101;    //RAM
    18:character<=7'd101;    //RAM
19:character<=7'd101;    //RAM
    20:character<=7'd101;    //RAM
    21:character<=7'd101;    //RAM
22:character<=7'd101;    //RAM
    23:character<=7'd101;    //RAM
    24:character<=7'd101;    //RAM
25:character<=7'd101;    //RAM
    default: character<=7'd32; //(space)
    endcase*/  //RAM


  23: case (horiz_char)
    52:character<=7'd83;    //S
    53:character<=7'd119;   //w
    54:character<=7'd105;   //i
55:character<=7'd116;   //t
56:character<=7'd99;    //c
57:character<=7'd104;   //h
    59:character<=7'd35;    //#
60:character<=7'd53;    //5
    61:character<=7'd58;    //:
    63:character<=7'd65;    //A
64:character<=7'd117;   //u
    65:character<=7'd116;   //t
    66:character<=7'd111;   //o
67:character<=7'd116;   //t
    68:character<=7'd117;   //u
69:character<=7'd110;   //n
70:character<=7'd101;   //e
    72:character<=7'd116;   //t
```

```
      73:character<=7'd111;   //o
      75:character<=7'd82;    //R
   76:character<=7'd97;    //a
   77:character<=7'd100;   //d
   78:character<=7'd105;   //i
      79:character<=7'd111;   //o
      81:character<=7'd83;    //S
      82:character<=7'd116;   //t
      83:character<=7'd97;    //a
   84:character<=7'd116;   //t
   85:character<=7'd105;   //i
   86:character<=7'd111;   //o
      87:character<=7'd110;   //n
      88:character<=7'd115;   //s
      default: character<=7'd32; //(space)
      endcase

    27: case (horiz_char)
      35:character<=7'd65;    //A
      36:character<=7'd77;    //M
      38:character<=7'd70;    //F
   39:character<=7'd114;   //r
   40:character<=7'd101;   //e
   41:character<=7'd113;   //q
      42:character<=7'd117;   //u
   43:character<=7'd101;   //e
      44:character<=7'd110;   //n
      45:character<=7'd99;    //c
   46:character<=7'd121;   //y
      48:character<=7'd66;    //B
      49:character<=7'd97;    //a
   50:character<=7'd110;   //n
      51:character<=7'd100;   //d
   53:character<=7'd40;    //(
   54:character<=7'd54;    //6
      55:character<=7'd48;    //0
      56:character<=7'd48;    //0
      57:character<=7'd45;    //-
   58:character<=7'd49;    //1
   59:character<=7'd52;    //4
   60:character<=7'd53;    //5
      61:character<=7'd48;    //0
      63:character<=7'd107;   //k
   64:character<=7'd72;    //H
   65:character<=7'd122;   //z
      66:character<=7'd41;    //)
      default: character<=7'd32; //(space)
      endcase
```

```
  29: case (horiz_char)
    42:character<=7'd70;    //F
    43:character<=7'd111;   //o
    44:character<=7'd117;   //u
45:character<=7'd114;   //r
46:character<=7'd105;   //i
47:character<=7'd101;   //e
    48:character<=7'd114;   //r
50:character<=7'd84;    //T
    51:character<=7'd114;   //r
    52:character<=7'd97;    //a
53:character<=7'd110;   //n
    54:character<=7'd115;   //s
    55:character<=7'd102;   //f
56:character<=7'd111;   //o
    57:character<=7'd114;   //r
58:character<=7'd109;   //m
default: character<=7'd32; //(space)
    endcase

  31: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  32: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  33: case (horiz_char)
    3:character<=7'd123;  //+
    default: character<=7'd32; //(space)
    endcase
  34: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  35: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  36: case (horiz_char)
    3:character<=7'd123;  //+
    default: character<=7'd32; //(space)
    endcase
  37: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
```

```
    endcase
  38: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  39: case (horiz_char)
    3:character<=7'd123;  //+
    default: character<=7'd32; //(space)
    endcase
  40: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  41: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  42: case (horiz_char)
    3:character<=7'd123;  //+
    default: character<=7'd32; //(space)
    endcase
  43: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  44: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  45: case (horiz_char)
    3:character<=7'd123;  //+
    default: character<=7'd32; //(space)
    endcase
  46: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  47: case (horiz_char)
    3:character<=7'd124;  //|
    default: character<=7'd32; //(space)
    endcase
  48: case (horiz_char)
    3:character<=7'd125;  //L
    4:character<=7'd127;  //_
    5:character<=7'd126;  //+
    6:character<=7'd127;  //_
    7:character<=7'd127;  //_
    8:character<=7'd127;  //_
```

```
9:character<=7'd127;  //_
10:character<=7'd126;  //+
11:character<=7'd127;  //_
12:character<=7'd127;  //_
13:character<=7'd127;  //_
14:character<=7'd127;  //_
15:character<=7'd126;  //+
16:character<=7'd127;  //_
17:character<=7'd127;  //_
18:character<=7'd127;  //_
19:character<=7'd127;  //_
20:character<=7'd126;  //+
21:character<=7'd127;  //_
22:character<=7'd127;  //_
23:character<=7'd127;  //_
24:character<=7'd127;  //_
25:character<=7'd126;  //+
26:character<=7'd127;  //_
27:character<=7'd127;  //_
28:character<=7'd127;  //_
29:character<=7'd127;  //_
30:character<=7'd126;  //+
31:character<=7'd127;  //_
32:character<=7'd127;  //_
33:character<=7'd127;  //_
34:character<=7'd127;  //_
35:character<=7'd126;  //+
31:character<=7'd127;  //_
32:character<=7'd127;  //_
33:character<=7'd127;  //_
34:character<=7'd127;  //_
35:character<=7'd126;  //+
31:character<=7'd127;  //_
32:character<=7'd127;  //_
33:character<=7'd127;  //_
34:character<=7'd127;  //_
35:character<=7'd126;  //+
31:character<=7'd127;  //_
32:character<=7'd127;  //_
33:character<=7'd127;  //_
34:character<=7'd127;  //_
35:character<=7'd126;  //+
16:character<=7'd127;  //_
17:character<=7'd127;  //_
18:character<=7'd127;  //_
19:character<=7'd127;  //_
20:character<=7'd126;  //+
21:character<=7'd127;  //_
```

```
22:character<=7'd127;  //_
23:character<=7'd127;  //_
24:character<=7'd127;  //_
25:character<=7'd126;  //+
26:character<=7'd127;  //_
27:character<=7'd127;  //_
28:character<=7'd127;  //_
29:character<=7'd127;  //_
30:character<=7'd126;  //+
31:character<=7'd127;  //_
32:character<=7'd127;  //_
33:character<=7'd127;  //_
34:character<=7'd127;  //_
35:character<=7'd126;  //+
36:character<=7'd127;  //_
37:character<=7'd127;  //_
38:character<=7'd127;  //_
39:character<=7'd127;  //_
40:character<=7'd126;  //+
41:character<=7'd127;  //_
42:character<=7'd127;  //_
43:character<=7'd127;  //_
44:character<=7'd127;  //_
45:character<=7'd126;  //+
46:character<=7'd127;  //_
47:character<=7'd127;  //_
48:character<=7'd127;  //_
49:character<=7'd127;  //_
50:character<=7'd126;  //+
51:character<=7'd127;  //_
52:character<=7'd127;  //_
53:character<=7'd127;  //_
54:character<=7'd127;  //_
55:character<=7'd126;  //+
56:character<=7'd127;  //_
57:character<=7'd127;  //_
58:character<=7'd127;  //_
59:character<=7'd127;  //_
60:character<=7'd126;  //+
61:character<=7'd127;  //_
62:character<=7'd127;  //_
63:character<=7'd127;  //_
64:character<=7'd127;  //_
65:character<=7'd126;  //+
66:character<=7'd127;  //_
67:character<=7'd127;  //_
68:character<=7'd127;  //_
69:character<=7'd127;  //_
```

```
    70:character<=7'd126;  //+
    71:character<=7'd127;  //_
    72:character<=7'd127;  //_
    73:character<=7'd127;  //_
    74:character<=7'd127;  //_
    75:character<=7'd126;  //+
    76:character<=7'd127;  //_
    77:character<=7'd127;  //_
    78:character<=7'd127;  //_
    79:character<=7'd127;  //_
    80:character<=7'd126;  //+
    81:character<=7'd127;  //_
    82:character<=7'd127;  //_
    83:character<=7'd127;  //_
    84:character<=7'd127;  //_
    85:character<=7'd126;  //+
    86:character<=7'd127;  //_
    87:character<=7'd127;  //_
    88:character<=7'd127;  //_
    89:character<=7'd127;  //_
    90:character<=7'd126;  //+
    91:character<=7'd127;  //_
    92:character<=7'd127;  //_
    93:character<=7'd127;  //_
    94:character<=7'd127;  //_
    default: character<=7'd32; //(space)
    endcase
  49: case (horiz_char)
        4:character<=7'd54;   //6
        5:character<=7'd48;       //0
        6:character<=7'd48;       //0
        9:character<=7'd54;       //6
        10:character<=7'd53;  //5
        11:character<=7'd48;  //0
        14:character<=7'd55;  //7
        15:character<=7'd48;  //0
        16:character<=7'd48;  //0
        19:character<=7'd55;  //7
        20:character<=7'd53;  //5
        21:character<=7'd48;  //0
        24:character<=7'd56;  //8
        25:character<=7'd48;  //0
        26:character<=7'd48;  //0
        29:character<=7'd56;  //8
        30:character<=7'd53;  //5
        31:character<=7'd48;  //0
        34:character<=7'd57;  //9
        35:character<=7'd48;  //0
```

```
36:character<=7'd48;  //0
39:character<=7'd57;  //9
40:character<=7'd53;  //5
41:character<=7'd48;  //0
44:character<=7'd49;  //1
45:character<=7'd48;  //0
46:character<=7'd48;  //0
47:character<=7'd48;  //0
49:character<=7'd49;  //1
50:character<=7'd48;  //0
51:character<=7'd53;  //5
52:character<=7'd48;  //0
54:character<=7'd49;  //1
55:character<=7'd49;  //1
56:character<=7'd48;  //0
57:character<=7'd48;  //0
59:character<=7'd49;  //1
60:character<=7'd49;  //1
61:character<=7'd53;  //5
62:character<=7'd48;  //0
64:character<=7'd49;  //1
65:character<=7'd50;  //2
66:character<=7'd48;  //0
67:character<=7'd48;  //0
69:character<=7'd49;  //1
70:character<=7'd50;  //2
71:character<=7'd53;  //5
72:character<=7'd48;  //0
74:character<=7'd49;  //1
75:character<=7'd51;  //3
76:character<=7'd48;  //0
77:character<=7'd48;  //0
79:character<=7'd49;  //1
80:character<=7'd51;  //3
81:character<=7'd53;  //5
82:character<=7'd48;  //0
84:character<=7'd49;  //1
85:character<=7'd52;  //4
86:character<=7'd48;  //0
87:character<=7'd48;  //0
89:character<=7'd49;  //1
90:character<=7'd52;  //4
91:character<=7'd53;  //5
92:character<=7'd48;  //0
94:character<=7'd40;  //(
95:character<=7'd107;  //k
96:character<=7'd72;  //H
97:character<=7'd122;  //z
```

```
                98:character<=7'd41;  //)
                default: character<=7'd32; //(space)
                endcase
            default: character<=7'd32;  //(space)
        endcase
        end

   always @ (posedge pixel_clock)
   pixel<=pixelrom[newhoriz_count4];


   always @(posedge pixel_clock)
    if (pixel) begin                    //font color -black
        if (vert_char > 30) begin
          vga_out_red<=8'h00;
        vga_out_green<=8'h00;
        vga_out_blue<=8'h00;
          end
        else begin
        vga_out_red<=8'h00;
        vga_out_green<=8'h00;
        vga_out_blue<=8'hCC;
          end
        end

     else begin                     //background color - gray
        vga_out_red<=8'hFF;
        vga_out_green<=8'hFF;
        vga_out_blue<=8'hFF;   end

  endmodule
//
**********************************************************************************************


//*********************************************************************************************
//Wajahat Khan
//*********************************************************************
module demodsampstr(clk, reset, data_in_ready, insample, data0, data1, data2, data3, data4, data5, data6, data7,
data8, data9,
                data10, data11, data12, data13, data14, data15, data16, data17, data18, data19,
                data20, data21, data22, data23, data24, data25, data26, data27, data28, data29);

input clk, reset, data_in_ready;
input [17:0] insample;
output [17:0] data0, data1, data2, data3, data4, data5, data6, data7, data8, data9,
                data10, data11, data12, data13, data14, data15, data16, data17, data18, data19,
                data20, data21, data22, data23, data24, data25, data26, data27, data28, data29;
//output [27:0] sample [4:0];
```

```
reg [17:0] data0, data1, data2, data3, data4, data5, data6, data7, data8, data9,
                data10, data11, data12, data13, data14, data15, data16, data17, data18, data19,
                data20, data21, data22, data23, data24, data25, data26, data27, data28, data29;



always @ (posedge clk)  //try working with negede to improve performance
begin
if (!reset)
        begin
    data0 <= 0;
        data1 <= 0;
        data2 <= 0;
        data3 <= 0;
        data4 <= 0;
        data5 <= 0;
        data6 <= 0;
        data7 <= 0;
        data8 <= 0;
        data9 <= 0;
        data10 <= 0;
        data11 <= 0;
        data12 <= 0;
        data13 <= 0;
        data14 <= 0;
        data15 <= 0;
        data16<= 0;
        data17 <= 0;
        data18 <= 0;
        data19 <= 0;
        data20 <= 0;
        data21 <= 0;
        data22 <= 0;
        data23 <= 0;
        data24 <= 0;
        data25 <= 0;
        data26 <= 0;
        data27 <= 0;
        data28 <= 0;
        data29 <= 0;
        end

else if (data_in_ready)
        begin
        data0 <= insample;
        data1 <= data0;
        data2 <= data1;
```

```
        data3 <= data2;
        data4 <= data3;
        data5 <= data4;
        data6 <= data5;
        data7 <= data6;
        data8 <= data7;
        data9 <= data8;
        data10 <= data9;
        data11 <= data10;
        data12 <= data11;
        data13 <= data12;
        data14 <= data13;
        data15 <= data14;
        data16 <= data15;
        data17 <= data16;
        data18 <= data17;
        data19 <= data18;
        data20 <= data19;
        data21 <= data20;
        data22 <= data21;
        data23 <= data22;
        data24 <= data23;
        data25 <= data24;
        data26 <= data25;
        data27 <= data26;
        data28 <= data27;
        data29 <= data28;
        end

else
        begin
        data0 <= data0;
        data1 <= data1;
        data2 <= data2;
        data3 <= data3;
        data4 <= data4;
        data5 <= data5;
        data6 <= data6;
        data7 <= data7;
        data8 <= data8;
        data9 <= data9;
        data10 <= data10;
        data11 <= data11;
        data12 <= data12;
        data13 <= data13;
        data14 <= data14;
        data15 <= data15;
        data16 <= data16;
```

```
            data17 <= data17;
            data18 <= data18;
            data19 <= data19;
            data20 <= data20;
            data21 <= data21;
            data22 <= data22;
            data23 <= data23;
            data24 <= data24;
            data25 <= data25;
            data26 <= data26;
            data27 <= data27;
            data28 <= data28;
            data29 <= data29;
            end

end //always@


endmodule
//****************************************************
module fiveinaccumulator(reset, clk, in1, in2, in3, in4, in5, in6, in7, in8, in9, in10,
                        in11, in12, in13, in14, in15, in16, in17 , in18, in19, in20,
                        in21, in22, in23, in24, in25, in26, in27, in28, in29, in30,
                        outen, d, output_accum, out_dsp);

input [35:0] in1;//, in2, in3, in4, in5, in6, in7, in8, in9, in10;
input [35:0] in2;
input [35:0] in3,in4,in5,in6,in7,in8,in9,in10;
input [35:0] in11, in12, in13, in14, in15, in16, in17 , in18, in19, in20,
  in21, in22, in23, in24, in25, in26, in27, in28, in29, in30;

input reset, clk, outen;

output [19:0] out_dsp, output_accum;
output [36:0] d;

//output [31:0] in1, in5;
//output accum_clk;
//reg accum_clk;
//reg [31:0] in1, in2, in3, in4, in5;

wire [36:0] d, signext_in1, signext_in2, signext_in3, signext_in4, signext_in5,
            signext_in6, signext_in7, signext_in8, signext_in9, signext_in10,
            signext_in11, signext_in12, signext_in13, signext_in14, signext_in15,
            signext_in16, signext_in17, signext_in18, signext_in19, signext_in20,
            signext_in21, signext_in22, signext_in23, signext_in24, signext_in25,
            signext_in26, signext_in27, signext_in28, signext_in29, signext_in30;
```

```
wire [19:0] output_accum;       //AC 97's codec's resolution is 18 bit, lower 2bits are zero
reg [19:0] out_dsp ;

always @ (posedge clk )
begin
  if (!reset)
  out_dsp <= 0;

  else begin
       if (outen)    //latch
         out_dsp <= output_accum;
       else
         out_dsp <= out_dsp;
       end

end


 //     accum_clk <= ~accum_clk;
/*
//always @ (posedge accum_clk)
     if (accum_clk) begin
       in1 =  z1[35:4];
       in2 =  z2[35:4];
       in3 =  z3[35:4];
       in4 =  z4[35:4];
       in5 =  z5[35:4];
       if (!reset_accum)
         q <= 0;
       else
         q <= d;


  //latch

  end
end

*/

assign signext_in1 = {in1[35], in1[35], in1[35], in1[35], in1[35], in1[35:4]} ;
assign signext_in2 = {in2[35], in2[35], in2[35], in2[35], in2[35], in2[35:4]} ;
assign signext_in3 = {in3[35], in3[35], in3[35], in3[35], in3[35], in3[35:4]} ;
assign signext_in4 = {in4[35], in4[35], in4[35], in4[35], in4[35], in4[35:4]} ;
assign signext_in5 = {in5[35], in5[35], in5[35], in5[35], in5[35], in5[35:4]} ;

assign signext_in6 = {in6[35], in6[35], in6[35], in6[35], in6[35], in6[35:4]} ;
assign signext_in7 = {in7[35], in7[35], in7[35], in7[35], in7[35], in7[35:4]} ;
```

```
assign signext_in8 = {in8[35], in8[35], in8[35], in8[35], in8[35], in8[35:4]} ;
assign signext_in9 = {in9[35], in9[35], in9[35], in9[35], in9[35], in9[35:4]} ;
assign signext_in10 = {in10[35], in10[35], in10[35], in10[35], in10[35], in10[35:4]} ;

assign signext_in11 = {in11[35], in11[35], in11[35], in11[35], in11[35], in11[35:4]} ;
assign signext_in12 = {in12[35], in12[35], in12[35], in12[35], in12[35], in12[35:4]} ;
assign signext_in13 = {in13[35], in13[35], in13[35], in13[35], in13[35], in13[35:4]} ;
assign signext_in14 = {in14[35], in14[35], in14[35], in14[35], in14[35], in14[35:4]} ;
assign signext_in15 = {in15[35], in15[35], in15[35], in15[35], in15[35], in15[35:4]} ;

assign signext_in16 = {in16[35], in16[35], in16[35], in16[35], in16[35], in16[35:4]} ;
assign signext_in17 = {in17[35], in17[35], in17[35], in17[35], in17[35], in17[35:4]} ;
assign signext_in18 = {in18[35], in18[35], in18[35], in18[35], in18[35], in18[35:4]} ;
assign signext_in19 = {in19[35], in19[35], in19[35], in19[35], in19[35], in19[35:4]} ;
assign signext_in20 = {in20[35], in20[35], in20[35], in20[35], in20[35], in20[35:4]} ;

assign signext_in21 = {in21[35], in21[35], in21[35], in21[35], in21[35], in21[35:4]} ;
assign signext_in22 = {in22[35], in22[35], in22[35], in22[35], in22[35], in22[35:4]} ;
assign signext_in23 = {in23[35], in23[35], in23[35], in23[35], in23[35], in23[35:4]} ;
assign signext_in24 = {in24[35], in24[35], in24[35], in24[35], in24[35], in24[35:4]} ;
assign signext_in25 = {in25[35], in25[35], in25[35], in25[35], in25[35], in25[35:4]} ;

assign signext_in26 = {in26[35], in26[35], in26[35], in26[35], in26[35], in26[35:4]} ;
assign signext_in27 = {in27[35], in27[35], in27[35], in27[35], in27[35], in27[35:4]} ;
assign signext_in28 = {in28[35], in28[35], in28[35], in28[35], in28[35], in28[35:4]} ;
assign signext_in29 = {in29[35], in29[35], in29[35], in29[35], in29[35], in29[35:4]} ;
assign signext_in30 = {in30[35], in30[35], in30[35], in30[35], in30[35], in30[35:4]} ;


assign d = signext_in1 + signext_in2 + signext_in3 + signext_in4 + signext_in5 +
        signext_in6 + signext_in7 + signext_in8 + signext_in9 + signext_in10 +
        signext_in11 + signext_in12 + signext_in13 + signext_in14 + signext_in15+
        signext_in16 + signext_in17 + signext_in18 + signext_in19 + signext_in20 +
        signext_in21 + signext_in22 + signext_in23 + signext_in24 + signext_in25 +
        signext_in26 + signext_in27 + signext_in28 + signext_in29 + signext_in30;


////////////////////////////////////////////////////////////////////////
assign output_accum = {d[36:19], 2'b00};              //?? which bits, find by experimentation
//which bits should we sacrifice, MSB's or LSB's
////////////////////////////////////////////

endmodule
//**********************************************************************************************

//5 input multiplier
//takes inputs in bin offset X, 2'scomplement and output in two's complement
//inputs Xn ,Yn , output Zn
```

```
module fiveinputmul
        (ysignmagn1, ysignmagn2, ysignmagn3, ysignmagn4, ysignmagn5, ysignmagn6, ysignmagn7, ysignmagn8,
ysignmagn9, ysignmagn10,
        ysignmagn11, ysignmagn12, ysignmagn13, ysignmagn14, ysignmagn15, ysignmagn16, ysignmagn17,
ysignmagn18, ysignmagn19, ysignmagn20,
        ysignmagn21, ysignmagn22, ysignmagn23, ysignmagn24, ysignmagn25, ysignmagn26, ysignmagn27,
ysignmagn28, ysignmagn29, ysignmagn30,
        z1, z2, z3, z4, z5, z6, z7, z8, z9, z10,
        z11, z12, z13, z14, z15, z16, z17, z18, z19, z20,
        z21, z22, z23, z24, z25, z26, z27, z28, z29, z30);

input [17:0] ysignmagn1, ysignmagn2, ysignmagn3, ysignmagn4, ysignmagn5, ysignmagn6, ysignmagn7,
ysignmagn8, ysignmagn9, ysignmagn10,
    ysignmagn11, ysignmagn12, ysignmagn13, ysignmagn14, ysignmagn15, ysignmagn16, ysignmagn17,
ysignmagn18, ysignmagn19, ysignmagn20,
    ysignmagn21, ysignmagn22, ysignmagn23, ysignmagn24, ysignmagn25, ysignmagn26, ysignmagn27,
ysignmagn28, ysignmagn29, ysignmagn30;




output [35:0] z1, z2, z3, z4, z5, z6, z7, z8, z9, z10,
    z11, z12, z13, z14, z15, z16, z17, z18, z19, z20,
    z21, z22, z23, z24, z25, z26, z27, z28, z29, z30;

wire [17:0] x1magn, x2magn, x3magn, x4magn, x5magn, x6magn, x7magn, x8magn, x9magn, x10magn,
    x11magn, x12magn, x13magn, x14magn, x15magn, x16magn, x17magn, x18magn, x19magn, x20magn,
    x21magn, x22magn, x23magn, x24magn, x25magn, x26magn, x27magn, x28magn, x29magn, x30magn;

wire [17:0] y1magn, y2magn, y3magn, y4magn, y5magn, y6magn, y7magn, y8magn, y9magn, y10magn,
    y11magn, y12magn, y13magn, y14magn, y15magn, y16magn, y17magn, y18magn, y19magn, y20magn,
    y21magn, y22magn, y23magn, y24magn, y25magn, y26magn, y27magn, y28magn, y29magn, y30magn;

wire [17:0] x1, x2, x3, x4, x5, x6, x7, x8, x9, x10,
    x11, x12, x13, x14, x15, x16, x17, x18, x19, x20,
    x21, x22, x23, x24, x25, x26, x27, x28, x29, x30;



wire [35:0] z1magn, z2magn, z3magn, z4magn, z5magn, z6magn, z7magn, z8magn, z9magn, z10magn,
    z11magn, z12magn, z13magn, z14magn, z15magn, z16magn, z17magn, z18magn, z19magn, z20magn,
    z21magn, z22magn, z23magn, z24magn, z25magn, z26magn, z27magn, z28magn, z29magn, z30magn;

//can we combine to x1 and xmagn to get better performance?

wire [17:0] xbinoffset1, xbinoffset2, xbinoffset3, xbinoffset4, xbinoffset5, xbinoffset6, xbinoffset7, xbinoffset8,
xbinoffset9, xbinoffset10,
```

```
        xbinoffset11, xbinoffset12, xbinoffset13, xbinoffset14, xbinoffset15, xbinoffset16, xbinoffset17, xbinoffset18,
xbinoffset19, xbinoffset20,
        xbinoffset21, xbinoffset22, xbinoffset23, xbinoffset24, xbinoffset25, xbinoffset26, xbinoffset27, xbinoffset28,
xbinoffset29, xbinoffset30;



assign xbinoffset1 = 18'd10475;
assign xbinoffset2 = 18'd11889;
assign xbinoffset3 = 18'd16057;
assign xbinoffset4 = 18'd22788;
assign xbinoffset5 = 18'd31770;

assign xbinoffset6 = 18'd42584;
assign xbinoffset7 = 18'd54727;
assign xbinoffset8 = 18'd67630;
assign xbinoffset9 = 18'd80691;
assign xbinoffset10 = 18'd93297;

assign xbinoffset11 = 18'd104860;
assign xbinoffset12 = 18'd114836;
assign xbinoffset13 = 18'd122759;
assign xbinoffset14 = 18'd128256;
assign xbinoffset15 = 18'd131071;

assign xbinoffset16 =  18'd131071;
assign xbinoffset17 =  18'd128256;
assign xbinoffset18 =  18'd122759;
assign xbinoffset19 =  18'd114836;
assign xbinoffset20 =  18'd104860;

assign xbinoffset21 =  18'd93297;
assign xbinoffset22 =  18'd80691;
assign xbinoffset23 =  18'd67630;
assign xbinoffset24 =  18'd54727;
assign xbinoffset25 =  18'd42584;

assign xbinoffset26 =  18'd31770;
assign xbinoffset27 =  18'd22788;
assign xbinoffset28 =  18'd16057;
assign xbinoffset29 =  18'd11889;
assign xbinoffset30 =  18'd10475;




assign x1 = {~xbinoffset1[17], xbinoffset1[16:0]};
```

```
assign x2 = {~xbinoffset2[17], xbinoffset2[16:0]};
assign x3 = {~xbinoffset3[17], xbinoffset3[16:0]};
assign x4 = {~xbinoffset4[17], xbinoffset4[16:0]};
assign x5 = {~xbinoffset5[17], xbinoffset5[16:0]};
assign x6 = {~xbinoffset6[17], xbinoffset6[16:0]};
assign x7 = {~xbinoffset7[17], xbinoffset7[16:0]};
assign x8 = {~xbinoffset8[17], xbinoffset8[16:0]};
assign x9 = {~xbinoffset9[17], xbinoffset9[16:0]};
assign x10 = {~xbinoffset10[17], xbinoffset10[16:0]};
assign x11 = {~xbinoffset11[17], xbinoffset11[16:0]};
assign x12 = {~xbinoffset12[17], xbinoffset12[16:0]};
assign x13 = {~xbinoffset13[17], xbinoffset13[16:0]};
assign x14 = {~xbinoffset14[17], xbinoffset14[16:0]};
assign x15 = {~xbinoffset15[17], xbinoffset15[16:0]};
assign x16 = {~xbinoffset16[17], xbinoffset16[16:0]};
assign x17 = {~xbinoffset17[17], xbinoffset17[16:0]};
assign x18 = {~xbinoffset18[17], xbinoffset18[16:0]};
assign x19 = {~xbinoffset19[17], xbinoffset19[16:0]};
assign x20 = {~xbinoffset20[17], xbinoffset20[16:0]};
assign x21 = {~xbinoffset21[17], xbinoffset21[16:0]};
assign x22 = {~xbinoffset22[17], xbinoffset22[16:0]};
assign x23 = {~xbinoffset23[17], xbinoffset23[16:0]};
assign x24 = {~xbinoffset24[17], xbinoffset24[16:0]};
assign x25 = {~xbinoffset25[17], xbinoffset25[16:0]};
assign x26 = {~xbinoffset26[17], xbinoffset26[16:0]};
assign x27 = {~xbinoffset27[17], xbinoffset27[16:0]};
assign x28 = {~xbinoffset28[17], xbinoffset28[16:0]};
assign x29 = {~xbinoffset29[17], xbinoffset29[16:0]};
assign x30 = {~xbinoffset30[17], xbinoffset30[16:0]};


//assign foo = bar*5;
// synthesis attribute mult_style of foo is KCM;

assign x1magn = x1[17] ? (~x1 + 1) : x1;
assign x2magn = x2[17] ? (~x2 + 1) : x2;
assign x3magn = x3[17] ? (~x3 + 1) : x3;
assign x4magn = x4[17] ? (~x4 + 1) : x4;
assign x5magn = x5[17] ? (~x5 + 1) : x5;
assign x6magn = x6[17] ? (~x6 + 1) : x6;
assign x7magn = x7[17] ? (~x7 + 1) : x7;
assign x8magn = x8[17] ? (~x8 + 1) : x8;
assign x9magn = x9[17] ? (~x9 + 1) : x9;
assign x10magn = x10[17] ? (~x10 + 1) : x10;
assign x11magn = x11[17] ? (~x11 + 1) : x11;
assign x12magn = x12[17] ? (~x12 + 1) : x12;
assign x13magn = x13[17] ? (~x13 + 1) : x13;
assign x14magn = x14[17] ? (~x14 + 1) : x14;
```

```
assign x15magn = x15[17] ? (~x15 + 1) : x15;
assign x16magn = x16[17] ? (~x16 + 1) : x16;
assign x17magn = x17[17] ? (~x17 + 1) : x17;
assign x18magn = x18[17] ? (~x18 + 1) : x18;
assign x19magn = x19[17] ? (~x19 + 1) : x19;
assign x20magn = x20[17] ? (~x20 + 1) : x20;
assign x21magn = x21[17] ? (~x21 + 1) : x21;
assign x22magn = x22[17] ? (~x22 + 1) : x22;
assign x23magn = x23[17] ? (~x23 + 1) : x23;
assign x24magn = x24[17] ? (~x24 + 1) : x24;
assign x25magn = x25[17] ? (~x25 + 1) : x25;
assign x26magn = x26[17] ? (~x26 + 1) : x26;
assign x27magn = x27[17] ? (~x27 + 1) : x27;
assign x28magn = x28[17] ? (~x28 + 1) : x28;
assign x29magn = x29[17] ? (~x29 + 1) : x29;
assign x30magn = x30[17] ? (~x30 + 1) : x30;


assign y1magn = ysignmagn1[17] ? {1'b0, ysignmagn1[16:0]} : ysignmagn1;
assign y2magn = ysignmagn2[17] ? {1'b0, ysignmagn2[16:0]} : ysignmagn2;
assign y3magn = ysignmagn3[17] ? {1'b0, ysignmagn3[16:0]} : ysignmagn3;
assign y4magn = ysignmagn4[17] ? {1'b0, ysignmagn4[16:0]} : ysignmagn4;
assign y5magn = ysignmagn5[17] ? {1'b0, ysignmagn5[16:0]} : ysignmagn5;
assign y6magn = ysignmagn6[17] ? {1'b0, ysignmagn6[16:0]} : ysignmagn6;
assign y7magn = ysignmagn7[17] ? {1'b0, ysignmagn7[16:0]} : ysignmagn7;
assign y8magn = ysignmagn8[17] ? {1'b0, ysignmagn8[16:0]} : ysignmagn8;
assign y9magn = ysignmagn9[17] ? {1'b0, ysignmagn9[16:0]} : ysignmagn9;
assign y10magn = ysignmagn10[17] ? {1'b0, ysignmagn10[16:0]} : ysignmagn10;
assign y11magn = ysignmagn11[17] ? {1'b0, ysignmagn11[16:0]} : ysignmagn11;
assign y12magn = ysignmagn12[17] ? {1'b0, ysignmagn12[16:0]} : ysignmagn12;
assign y13magn = ysignmagn13[17] ? {1'b0, ysignmagn13[16:0]} : ysignmagn13;
assign y14magn = ysignmagn14[17] ? {1'b0, ysignmagn14[16:0]} : ysignmagn14;
assign y15magn = ysignmagn15[17] ? {1'b0, ysignmagn15[16:0]} : ysignmagn15;
assign y16magn = ysignmagn16[17] ? {1'b0, ysignmagn16[16:0]} : ysignmagn16;
assign y17magn = ysignmagn17[17] ? {1'b0, ysignmagn17[16:0]} : ysignmagn17;
assign y18magn = ysignmagn18[17] ? {1'b0, ysignmagn18[16:0]} : ysignmagn18;
assign y19magn = ysignmagn19[17] ? {1'b0, ysignmagn19[16:0]} : ysignmagn19;
assign y20magn = ysignmagn20[17] ? {1'b0, ysignmagn20[16:0]} : ysignmagn20;
assign y21magn = ysignmagn21[17] ? {1'b0, ysignmagn21[16:0]} : ysignmagn21;
assign y22magn = ysignmagn22[17] ? {1'b0, ysignmagn22[16:0]} : ysignmagn22;
assign y23magn = ysignmagn23[17] ? {1'b0, ysignmagn23[16:0]} : ysignmagn23;
assign y24magn = ysignmagn24[17] ? {1'b0, ysignmagn24[16:0]} : ysignmagn24;
assign y25magn = ysignmagn25[17] ? {1'b0, ysignmagn25[16:0]} : ysignmagn25;
assign y26magn = ysignmagn26[17] ? {1'b0, ysignmagn26[16:0]} : ysignmagn26;
assign y27magn = ysignmagn27[17] ? {1'b0, ysignmagn27[16:0]} : ysignmagn27;
assign y28magn = ysignmagn28[17] ? {1'b0, ysignmagn28[16:0]} : ysignmagn28;
assign y29magn = ysignmagn29[17] ? {1'b0, ysignmagn29[16:0]} : ysignmagn29;
assign y30magn = ysignmagn30[17] ? {1'b0, ysignmagn30[16:0]} : ysignmagn30;
```

```
assign z1magn = x1magn*y1magn;
assign z2magn = x2magn*y2magn;
assign z3magn = x3magn*y3magn;
assign z4magn = x4magn*y4magn;
assign z5magn = x5magn*y5magn;
assign z6magn = x6magn*y6magn;
assign z7magn = x7magn*y7magn;
assign z8magn = x8magn*y8magn;
assign z9magn = x9magn*y9magn;
assign z10magn = x10magn*y10magn;
assign z11magn = x11magn*y11magn;
assign z12magn = x12magn*y12magn;
assign z13magn = x13magn*y13magn;
assign z14magn = x14magn*y14magn;
assign z15magn = x15magn*y15magn;
assign z16magn = x16magn*y16magn;
assign z17magn = x17magn*y17magn;
assign z18magn = x18magn*y18magn;
assign z19magn = x19magn*y19magn;
assign z20magn = x20magn*y20magn;
assign z21magn = x21magn*y21magn;
assign z22magn = x22magn*y22magn;
assign z23magn = x23magn*y23magn;
assign z24magn = x24magn*y24magn;
assign z25magn = x25magn*y25magn;
assign z26magn = x26magn*y26magn;
assign z27magn = x27magn*y27magn;
assign z28magn = x28magn*y28magn;
assign z29magn = x29magn*y29magn;
assign z30magn = x30magn*y30magn;




assign z1 = (x1[17] ^ ysignmagn1[17]) ? (~z1magn + 1) : z1magn;
assign z2 = (x2[17] ^ ysignmagn2[17]) ? (~z2magn + 1) : z2magn;
assign z3 = (x3[17] ^ ysignmagn3[17]) ? (~z3magn + 1) : z3magn;
assign z4 = (x4[17] ^ ysignmagn4[17]) ? (~z4magn + 1) : z4magn;
assign z5 = (x5[17] ^ ysignmagn5[17]) ? (~z5magn + 1) : z5magn;
assign z6 = (x6[17] ^ ysignmagn6[17]) ? (~z6magn + 1) : z6magn;
assign z7 = (x7[17] ^ ysignmagn7[17]) ? (~z7magn + 1) : z7magn;
assign z8 = (x8[17] ^ ysignmagn8[17]) ? (~z8magn + 1) : z8magn;
assign z9 = (x9[17] ^ ysignmagn9[17]) ? (~z9magn + 1) : z9magn;
assign z10 = (x10[17] ^ ysignmagn10[17]) ? (~z10magn + 1) : z10magn;
assign z11 = (x11[17] ^ ysignmagn11[17]) ? (~z11magn + 1) : z11magn;
```

```
assign z12 = (x12[17] ^ ysignmagn12[17]) ? (~z12magn + 1) : z12magn;
assign z13 = (x13[17] ^ ysignmagn13[17]) ? (~z13magn + 1) : z13magn;
assign z14 = (x14[17] ^ ysignmagn14[17]) ? (~z14magn + 1) : z14magn;
assign z15 = (x15[17] ^ ysignmagn15[17]) ? (~z15magn + 1) : z15magn;
assign z16 = (x16[17] ^ ysignmagn16[17]) ? (~z16magn + 1) : z16magn;
assign z17 = (x17[17] ^ ysignmagn17[17]) ? (~z17magn + 1) : z17magn;
assign z18 = (x18[17] ^ ysignmagn18[17]) ? (~z18magn + 1) : z18magn;
assign z19 = (x19[17] ^ ysignmagn19[17]) ? (~z19magn + 1) : z19magn;
assign z20 = (x20[17] ^ ysignmagn20[17]) ? (~z20magn + 1) : z20magn;
assign z21 = (x21[17] ^ ysignmagn21[17]) ? (~z21magn + 1) : z21magn;
assign z22 = (x22[17] ^ ysignmagn22[17]) ? (~z22magn + 1) : z22magn;
assign z23 = (x23[17] ^ ysignmagn23[17]) ? (~z23magn + 1) : z23magn;
assign z24 = (x24[17] ^ ysignmagn24[17]) ? (~z24magn + 1) : z24magn;
assign z25 = (x25[17] ^ ysignmagn25[17]) ? (~z25magn + 1) : z25magn;
assign z26 = (x26[17] ^ ysignmagn26[17]) ? (~z26magn + 1) : z26magn;
assign z27 = (x27[17] ^ ysignmagn27[17]) ? (~z27magn + 1) : z27magn;
assign z28 = (x28[17] ^ ysignmagn28[17]) ? (~z28magn + 1) : z28magn;
assign z29 = (x29[17] ^ ysignmagn29[17]) ? (~z29magn + 1) : z29magn;
assign z30 = (x30[17] ^ ysignmagn30[17]) ? (~z30magn + 1) : z30magn;



endmodule

//**********************************************************************

module fsm2(clk, reset, data_inready, dcrom_add, outen, we, state, next);
//figure out the addressing for lpfrom write lpfrom_add as a function of outsel


input clk, data_inready, reset;
output [9:0] dcrom_add ;
output outen, we;
output [4:0] state, next;

reg inc_dcrom;
reg [4:0] state, next;
reg [9:0] dcrom_add;
reg outen, we;

parameter reset1 = 0;
parameter wait_data_in = 1;
parameter str_mod_sample = 2;
parameter buffer = 3;
parameter dc1 = 4;
parameter dc1ext = 5;
parameter dc2 = 6;
parameter dc2ext = 7;
```

```
parameter dc3 = 8;
parameter dc3ext = 9;
parameter dc4 = 10;
parameter dc4ext = 11;
parameter dc5 = 12;
parameter dc5ext = 13;
parameter dc6 = 14;
parameter dc6ext = 15;
parameter outputsample = 16;
parameter outputsample2 = 17;



always @ (posedge clk) begin

      if (!reset) begin
    state <= reset1;
    dcrom_add <= 0;
  end
  else begin
    state <= next;
    if (inc_dcrom)
      dcrom_add <= dcrom_add + 1;
    else dcrom_add <= dcrom_add;
  end

end




always @ (state or data_inready)
begin

case (state)

reset1:       begin

                  outen = 0;
                  we = 0;
                  next = wait_data_in;
      inc_dcrom = 0;
                  end

wait_data_in:
                  begin
                  outen = 0;
```

```
	inc_dcrom = 0;
	if (data_inready)
			next = str_mod_sample;
		else
			next = wait_data_in;
		end


str_mod_sample:
		begin
		we = 1;
		next = buffer;
		end

buffer:
	begin
	we = 0;
	next = dc1;
	inc_dcrom = 0;
	end



dc1:
		begin
	we = 0;
	inc_dcrom = 0;
		next = dc1ext;
		end


dc1ext:  begin
	next = dc2;
	end

dc2:	begin
	next = dc2ext;
		end


dc2ext:  begin
	next = dc3;
	end

dc3:	begin
		next = dc3ext;
		end
```

```
dc3ext:  begin
        next = dc4;
       end

dc4:          begin
                  next = dc4ext;
                  end

dc4ext:  begin
        next = dc5;
        end

dc5:          begin
        next = dc5ext;
                  end

dc5ext:  begin
        next = dc6;
        end

dc6:          begin
                  next = dc6ext;
                  end

dc6ext:  begin
        next = outputsample;
        end

outputsample:
                  begin
        outen = 1;
                  next = wait_data_in;
        inc_dcrom = 1;
                  end

default: begin
        next = reset1;
        end


endcase

end

endmodule


//**********************************************************************************
```

```
module multiplier(xbinoffset, y, zsignmagn);
input [13:0] xbinoffset, y;
output [17:0] zsignmagn;
wire [13:0] ymagn, xmagn, x;
wire [27:0] zmagn;
//wire [17:0] z;

assign x = {~xbinoffset[13], xbinoffset[12:0]};

assign xmagn = x[13] ? (~x + 1) : x;
assign ymagn = y[13] ? (~y + 1) : y;

assign zmagn = xmagn*ymagn;

assign zsignmagn = {(x[13] ^ y[13]), zmagn[27:11]}; //come back and see if if 27 is alwasy zero because it 27 and 26 bot store sign
endmodule
//**********************************************************************
/**********************************************************************
*    This file is owned and controlled by Xilinx and must be used     *
*    solely for design, simulation, implementation and creation of    *
*    design files limited to Xilinx devices or technologies. Use      *
*    with non-Xilinx devices or technologies is expressly prohibited  *
*    and immediately terminates your license.                         *
*                                                                     *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"    *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR          *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION  *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION      *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS        *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,          *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY         *
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE          *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR   *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF  *
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS  *
*    FOR A PARTICULAR PURPOSE.                                        *
*                                                                     *
*    Xilinx products are not intended for use in life support         *
*    appliances, devices, or systems. Use in such applications are    *
*    expressly prohibited.                                            *
*                                                                     *
*    (c) Copyright 1995-2004 Xilinx, Inc.                             *
*    All rights reserved.                                             *
**********************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
```

```
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file romdc.v when simulating
// the core, romdc. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Guide".

module romdc (
	addr,
	clk,
	dout);	// synthesis black_box

input [10 : 0] addr;
input clk;
output [13 : 0] dout;

// synopsys translate_off

	BLKMEMSP_V6_1 #(
		11,	// c_addr_width
		"0",	// c_default_data
		2048,	// c_depth
		0,	// c_enable_rlocs
		0,	// c_has_default_data
		0,	// c_has_din
		0,	// c_has_en
		0,	// c_has_limit_data_pitch
		0,	// c_has_nd
		0,	// c_has_rdy
		0,	// c_has_rfd
		0,	// c_has_sinit
		0,	// c_has_we
		18,	// c_limit_data_pitch
		"romdc.mif",	// c_mem_init_file
		0,	// c_pipe_stages
		0,	// c_reg_inputs
		"0",	// c_sinit_value
		14,	// c_width
		0,	// c_write_mode
		"0",	// c_ybottom_addr
		1,	// c_yclk_is_rising
		1,	// c_yen_is_high
		"hierarchy1",	// c_yhierarchy
		0,	// c_ymake_bmm
		"16kx1",	// c_yprimitive_type
		1,	// c_ysinit_is_high
		"1024",	// c_ytop_addr
		0,	// c_yuse_single_primitive
```

```
            1,      // c_ywe_is_high
            1)      // c_yydisable_warnings
        inst (
            .ADDR(addr),
            .CLK(clk),
            .DOUT(dout),
            .DIN(),
            .EN(),
            .ND(),
            .RFD(),
            .RDY(),
            .SINIT(),
            .WE());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of romdc is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of romdc is "black_box"

endmodule

//*****************************************************************************
//reset is active low
//data_inready takes in 14 bit 2's complement value
// data_inready takes in the data ready signal
// out_dsp is 20 bit twos compliment value with zeros as the two LSB's in order to be compatible with the AC 97
codec input format
//What bits of the final output are selected can be controlled by changing the accumulator module. Look
// for the statement enclosed in comments towards the end of the file. (output_accum = {q[72:55], 2'b00};)
// change the bit range by changing the rane in the square brackets. The range selected should be 18bit long.

//dont worry about the state, next, q, output_accum signals. Those were used for debugging purposes.



module topdsp(clk, reset, data_inready, data_in, freq, state, next, dcrom_add, dout, zsignmagn, we, dmoddata1,
dmoddata5,
            z1, z5, d, output_accum, outen, out_dsp);

input clk, reset, data_inready;
input  [13:0] data_in;
input freq;
```

```
output [4:0] state, next;
output [9:0] dcrom_add;
output [13:0] dout;
output [17:0] zsignmagn, dmoddata1, dmoddata5 ;
output [35:0] z1,z5;
output [36:0] d;
output [19:0]  output_accum, out_dsp;
output we, outen;


//output [31:0] accumin1, accumin5;
//output [31:0] in1, in5;

//output [2:0] outsel;
//output accum_clk;


reg data_inready1, data_inready2, data_inready3, data_inready4;

wire [9:0] dcrom_add;

//wire [2:0] outsel;
wire [17:0]  dmoddata0,
        dmoddata1, dmoddata2, dmoddata3, dmoddata4, dmoddata5,
        dmoddata6, dmoddata7, dmoddata8, dmoddata9,
                dmoddata10, dmoddata11, dmoddata12, dmoddata13, dmoddata14,
        dmoddata15, dmoddata16, dmoddata17, dmoddata18, dmoddata19,
                dmoddata20, dmoddata21, dmoddata22, dmoddata23, dmoddata24,
        dmoddata25, dmoddata26, dmoddata27, dmoddata28, dmoddata29, zsignmagn;

wire [35:0] z1, z2, z3, z4, z5, z6, z7, z8, z9, z10,
        z11, z12, z13, z14, z15, z16, z17, z18, z19, z20,
        z21, z22, z23, z24, z25, z26, z27, z28, z29, z30;

wire outen, we;
wire [4:0] state, next;


//reg [31:0] accumin1, accumin2, accumin3, accumin4, accumin5;


always @ (posedge clk)
  begin
  data_inready1 <= data_inready;
  data_inready2 <= data_inready1;
  data_inready3 <= data_inready2;
  data_inready4 <= data_inready3;
```

```
  end
/*
begin
  accumin1 =  z1[35:4];
  accumin2 =  z2[35:4];
  accumin3 =  z3[35:4];
  accumin4 =  z4[35:4];
  accumin5 =  z5[35:4];
end

 */

romdc dcrom (.addr({freq,dcrom_add}), .clk(clk), .dout(dout));
//foo dcrom (.addr(dcrom_add), .clk(clk), .dout(dout));
//assign dout = 14'd4893;
//assign dout = 14'b1000;


fsm2 finite (.clk(clk), .reset(reset), .data_inready(data_inready4), .dcrom_add(dcrom_add),
                        .outen(outen), .we(we), .state(state), .next(next)) ;


multiplier mul2(.xbinoffset(dout), .y(data_in), .zsignmagn(zsignmagn));

//rom_28x30 lpfroma (.addr(outsel), .data1(data1), .data2(data2), .data3(data3), .data4(data4), .data5(data5));


demodsampstr dmodsmpstr (clk, reset, we, zsignmagn, dmoddata0, dmoddata1, dmoddata2, dmoddata3, dmoddata4,
dmoddata5, dmoddata6, dmoddata7, dmoddata8, dmoddata9,
                dmoddata10, dmoddata11, dmoddata12, dmoddata13, dmoddata14, dmoddata15, dmoddata16,
dmoddata17, dmoddata18, dmoddata19,
                dmoddata20, dmoddata21, dmoddata22, dmoddata23, dmoddata24, dmoddata25, dmoddata26,
dmoddata27, dmoddata28, dmoddata29);



fiveinputmul mul (dmoddata0, dmoddata1, dmoddata2, dmoddata3, dmoddata4,
            dmoddata5, dmoddata6, dmoddata7, dmoddata8, dmoddata9,
                        dmoddata10, dmoddata11, dmoddata12, dmoddata13, dmoddata14,
            dmoddata15, dmoddata16, dmoddata17, dmoddata18, dmoddata19,
                        dmoddata20, dmoddata21, dmoddata22, dmoddata23, dmoddata24,
            dmoddata25, dmoddata26, dmoddata27, dmoddata28, dmoddata29,
                            z1, z2, z3, z4, z5, z6, z7, z8, z9, z10,
            z11, z12, z13, z14, z15, z16, z17, z18, z19, z20,
            z21, z22, z23, z24, z25, z26, z27, z28, z29, z30);


fiveinaccumulator accum (reset, clk,
```

```
            z1, z2, z3, z4, z5, z6, z7, z8, z9, z10,
            z11, z12, z13, z14, z15, z16, z17, z18, z19, z20,
            z21, z22, z23, z24, z25, z26, z27, z28, z29, z30,
            outen, d, output_accum, out_dsp);



//synthesis attribute period of clk is 36ns;
//sythesis attribute of clk is true;

endmodule

//*****************************************************************************
//Ebad Ahmed
//*****************************************************************************
/*****************************************************************************
*    This file is owned and controlled by Xilinx and must be used           *
*    solely for design, simulation, implementation and creation of          *
*    design files limited to Xilinx devices or technologies. Use            *
*    with non-Xilinx devices or technologies is expressly prohibited        *
*    and immediately terminates your license.                               *
*                                                                           *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"          *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR                *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION        *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION            *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS              *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,                *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE       *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY               *
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE                *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR         *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF        *
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        *
*    FOR A PARTICULAR PURPOSE.                                              *
*                                                                           *
*    Xilinx products are not intended for use in life support               *
*    appliances, devices, or systems. Use in such applications are          *
*    expressly prohibited.                                                  *
*                                                                           *
*    (c) Copyright 1995-2004 Xilinx, Inc.                                   *
*    All rights reserved.                                                   *
*****************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file adder.v when simulating
```

```
// the core, adder. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Guide".

module adder (
	A,
	B,
	Q,
	CLK);	// synthesis black_box

input [15 : 0] A;
input [15 : 0] B;
output [15 : 0] Q;
input CLK;

// synopsys translate_off

	C_ADDSUB_V7_0 #(
		0,	// c_add_mode
		"0000", // c_ainit_val
		1,	// c_a_type
		16,	 // c_a_width
		0,	// c_bypass_enable
		0,	// c_bypass_low
		0,	// c_b_constant
		1,	// c_b_type
		"0",	 // c_b_value
		16,	 // c_b_width
		0,	// c_enable_rlocs
		0,	// c_has_aclr
		0,	// c_has_add
		0,	// c_has_ainit
		0,	// c_has_aset
		0,	// c_has_a_signed
		0,	// c_has_bypass
		0,	// c_has_bypass_with_cin
		0,	// c_has_b_in
		0,	// c_has_b_out
		0,	// c_has_b_signed
		0,	// c_has_ce
		0,	// c_has_c_in
		0,	// c_has_c_out
		0,	// c_has_ovfl
		1,	// c_has_q
		0,	// c_has_q_b_out
		0,	// c_has_q_c_out
		0,	// c_has_q_ovfl
		0,	// c_has_s
```

```
            0,      // c_has_sclr
            0,      // c_has_sinit
            0,      // c_has_sset
            15,     // c_high_bit
            1,      // c_latency
            0,      // c_low_bit
            16,     // c_out_width
            1,      // c_pipe_stages
            "0",    // c_sinit_val
            0,      // c_sync_enable
            1)      // c_sync_priority
        inst (
            .A(A),
            .B(B),
            .Q(Q),
            .CLK(CLK),
            .ACLR(),
            .ADD(),
            .AINIT(),
            .ASET(),
            .A_SIGNED(),
            .B_OUT(),
            .C_IN(),
            .B_SIGNED(),
            .C_OUT(),
            .B_IN(),
            .BYPASS(),
            .CE(),
            .OVFL(),
            .Q_C_OUT(),
            .Q_B_OUT(),
            .Q_OVFL(),
            .S(),
            .SCLR(),
            .SINIT(),
            .SSET());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of adder is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of adder is "black_box"
```

```
endmodule

//*********************************************************
module divider(clk, Reset_Sync, enable);

   input clk, Reset_Sync;
   output enable;
   reg enable;
   reg [24:0] Q;

   always @ (posedge clk)
   begin
      if (!Reset_Sync) Q <= 0;
      else if (Q == 25'd54999999)
         begin
           Q <= 0;
           enable <= 1;
          end
      else begin Q <= Q + 1;
             enable <= 0;
        end
   end

endmodule
//***********************************************
module timer(clk, Reset_Sync, enable, start_timer, expired);

input clk, enable, start_timer, Reset_Sync;
output expired;
reg [3:0] Q;
reg expired;

always @ (posedge clk)
begin
   if (!Reset_Sync) Q <= 4'b0001;
       else if(expired == 1) expired <= 0;
   else if (enable && start_timer) begin
            if (Q == 6) begin
                  Q <= 4'b0001;
              expired <= 1; end
            else begin
            Q <= Q + 1;
         expired <= 0;   end
      end
end

endmodule
//***********************************************
```

```
`timescale 1ns/1ps

module autofsm(clk, reset, start_auto, busy_auto, v_sync, start, scale_sch,
            scale_sch_we, fwd_inv, fwd_inv_we, fft_done, busy_write, busy_display,
            scan, freq3, data_in, pressed2, WE_fft);

   input clk, reset, start_auto, v_sync, fft_done, busy_display, scan, pressed2;
   input [15:0] data_in;
   output busy_auto, start, scale_sch_we, fwd_inv, fwd_inv_we, busy_write, freq3,
   WE_fft;
   output [9:0] scale_sch;
   //output [2:0] state, next;
   //output [10:0] count;

   reg busy_auto, start, scale_sch_we, fwd_inv, fwd_inv_we, busy_write, freq3,
   WE_fft, enable;
   reg [2:0] state, next;
   reg [9:0] scale_sch;
   reg [9:0] scan_count;
   reg [10:0] count;

   parameter IDLE = 0;
   parameter WAIT_VSYNC = 1;
   parameter COMPUTE_FFT = 2;
   parameter UNLOAD = 3;
   parameter WAIT_DISPLAY1 = 4;
   parameter WAIT_DISPLAY2 = 5;
   parameter TUNE = 6;
   parameter PLAY_STATION =  7;


   always @ (posedge clk) begin
     if(!reset) state <= IDLE;
     else state <= next;

     if(state == IDLE) begin
     count <= 0;
     scan_count <= 0;
     end
     else if(state == UNLOAD && count != 11'd1027) begin
     count <= count + 1;
     scan_count <= scan_count;
       enable <= 0;
     end
       else if(state == UNLOAD && count == 11'd1027) begin
       enable <= 1;
       count <= -1;
       end
```

```
   else if(state == TUNE) begin
   count <= count;
   scan_count <= scan_count + 1;
   end
   else begin
      count <= count;
      scan_count <= scan_count;
   end

end


 always @ (state or start_auto or fft_done or v_sync or busy_display or enable) begin

    busy_auto = 1; start = 0; scale_sch = 10'b1010101011; fwd_inv = 1;
    fwd_inv_we = 1; busy_write = 0; WE_fft = 0; // defaults

    case(state)

       IDLE: begin
            busy_auto = 0;
            if(start_auto) next = WAIT_VSYNC;
            else next = IDLE;
         end


       WAIT_VSYNC: if(!v_sync) begin
                 next = COMPUTE_FFT;
                 start = 1;
                          scale_sch_we = 1;
                 end
                 else next = WAIT_VSYNC;


       COMPUTE_FFT: if (fft_done) next = UNLOAD;
                  else next = COMPUTE_FFT;


       UNLOAD: begin
              WE_fft = 1;
                       busy_write = 1;
              if(enable) next = WAIT_DISPLAY1;
              else next = UNLOAD;
           end


       WAIT_DISPLAY1: if(!busy_display) next = WAIT_DISPLAY1;
                    else next = WAIT_DISPLAY2;
```

```
        WAIT_DISPLAY2: if(busy_display) next = WAIT_DISPLAY2;
                    else next = TUNE;


        TUNE: if(pressed2) next = PLAY_STATION;
            else next = IDLE;


        PLAY_STATION: begin
                                    if (data_in[15:11] == 5'b00000) next = TUNE;
                    else begin
                       freq3 = 1; // change!
                       if(scan && pressed2) next = TUNE;
                       else if(pressed2) next = PLAY_STATION;
                       else next = IDLE;
                    end
                end

                default: next = IDLE;

            endcase
        end
  endmodule
//******************************************************
 module bookfsm(clk, reset, expired, button, pressed, start_book, busy_book, reset_timer, WE_book, data_w, data_in,
address, freq_out);

input clk, reset, expired, start_book, button, pressed;
input [3:0] data_in;
output busy_book, reset_timer, WE_book, address, freq_out;
output [3:0] data_w;

reg WE_book, reset_timer, busy_book, start_timer, address, freq_out;
reg [3:0] data_w;
reg [1:0] state, next;

parameter IDLE = 0;
parameter READ = 1;
parameter WAIT = 2;
parameter WRITE = 3;

always @ (posedge clk or negedge reset) begin
   if(!reset) state <= IDLE;
   else state <= next;
   end
```

```
always @ (state or start_book or expired) begin

   WE_book = 1; reset_timer = 1; busy_book = 1; start_timer = 0; freq_out = 0; data_w = 0;
   address = 0;// defaults

   case(state)

      IDLE: begin
         busy_book = 0;
         if((start_book == 1) && (pressed == 1)) next = READ;
         else next = IDLE;
         end


      READ: begin
            next = WAIT;
            if(button == 0) address = 0;
            else address = 1;
            end


      WAIT: begin
           //if(data_in == 4'b0) next = IDLE;
           if((data_in == 4'b0110) && button == 0) begin
               freq_out = 0;
               next = IDLE;
           end
           else if((data_in == 4'b1001) && button == 1) begin
              freq_out = 1;
              next = IDLE;
           end
           else begin
               reset_timer = 0;
               start_timer = 1;
               if(pressed == 0) next = IDLE;
               else if((expired == 1) && (pressed == 1)) begin
               next = WRITE;
               start_timer = 0;
               reset_timer = 1;
              end
               else next = WAIT;
           end
       end


      WRITE: begin
            next = IDLE;
            WE_book = 0;
```

```
            if(button == 1) begin
               address = 1;
               data_w = 1001;
            end

            else begin
               address = 0;
               data_w = 0110;
            end
         end

      default: next = IDLE;

endcase
end
endmodule
//***********************************************
/*****************************************************************************
*    This file is owned and controlled by Xilinx and must be used           *
*    solely for design, simulation, implementation and creation of          *
*    design files limited to Xilinx devices or technologies. Use            *
*    with non-Xilinx devices or technologies is expressly prohibited        *
*    and immediately terminates your license.                               *
*                                                                           *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"          *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR                *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION        *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION            *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS              *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,                *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE       *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY               *
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE                *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR         *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF        *
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        *
*    FOR A PARTICULAR PURPOSE.                                              *
*                                                                           *
*    Xilinx products are not intended for use in life support               *
*    appliances, devices, or systems. Use in such applications are          *
*    expressly prohibited.                                                  *
*                                                                           *
*    (c) Copyright 1995-2004 Xilinx, Inc.                                   *
*    All rights reserved.                                                   *
*****************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).
```

```
// You must compile the wrapper file bookram.v when simulating
// the core, bookram. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Guide".

module bookram (
	addr,
	clk,
	din,
	dout,
	we);	// synthesis black_box

input [0 : 0] addr;
input clk;
input [3 : 0] din;
output [3 : 0] dout;
input we;

// synopsys translate_off

	BLKMEMSP_V6_1 #(
		1,	// c_addr_width
		"0",	// c_default_data
		2,	// c_depth
		0,	// c_enable_rlocs
		1,	// c_has_default_data
		1,	// c_has_din
		0,	// c_has_en
		0,	// c_has_limit_data_pitch
		0,	// c_has_nd
		0,	// c_has_rdy
		0,	// c_has_rfd
		0,	// c_has_sinit
		1,	// c_has_we
		18,	// c_limit_data_pitch
		"mif_file_16_1",	// c_mem_init_file
		0,	// c_pipe_stages
		0,	// c_reg_inputs
		"0",	// c_sinit_value
		4,	// c_width
		0,	// c_write_mode
		"0",	// c_ybottom_addr
		1,	// c_yclk_is_rising
		1,	// c_yen_is_high
		"hierarchy1",	// c_yhierarchy
		0,	// c_ymake_bmm
		"16kx1",	// c_yprimitive_type
```

```
        1,      // c_ysinit_is_high
        "1024", // c_ytop_addr
        0,      // c_yuse_single_primitive
        1,      // c_ywe_is_high
        1)      // c_yydisable_warnings
    inst (
        .ADDR(addr),
        .CLK(clk),
        .DIN(din),
        .DOUT(dout),
        .WE(we),
        .EN(),
        .ND(),
        .RFD(),
        .RDY(),
        .SINIT());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of bookram is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of bookram is "black_box"

endmodule
//***************************************************************
  /*****************************************************************************
*    This file is owned and controlled by Xilinx and must be used           *
*    solely for design, simulation, implementation and creation of          *
*    design files limited to Xilinx devices or technologies. Use            *
*    with non-Xilinx devices or technologies is expressly prohibited        *
*    and immediately terminates your license.                               *
*                                                                           *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"          *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR                *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION        *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION            *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS              *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,                *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE       *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY               *
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE                *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR         *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF        *
```

```
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        *
*    FOR A PARTICULAR PURPOSE.                                              *
*                                                                           *
*    Xilinx products are not intended for use in life support               *
*    appliances, devices, or systems. Use in such applications are          *
*    expressly prohibited.                                                  *
*                                                                           *
*    (c) Copyright 1995-2004 Xilinx, Inc.                                   *
*    All rights reserved.                                                   *
*****************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file fft.v when simulating
// the core, fft. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Guide".

module fft (
        xn_re,
        xn_im,
        start,
        fwd_inv,
        fwd_inv_we,
        scale_sch,
        scale_sch_we,
        clk,
        xk_re,
        xk_im,
        xn_index,
        xk_index,
        rfd,
        busy,
        dv,
        edone,
        done);    // synthesis black_box

input [15 : 0] xn_re;
input [15 : 0] xn_im;
input start;
input fwd_inv;
input fwd_inv_we;
input [9 : 0] scale_sch;
input scale_sch_we;
input clk;
output [15 : 0] xk_re;
output [15 : 0] xk_im;
```

```
output [9 : 0] xn_index;
output [9 : 0] xk_index;
output rfd;
output busy;
output dv;
output edone;
output done;

// synopsys translate_off

	XFFT_V3_0 #(
		3,	// c_arch
		1,	// c_bram_stages
		1,	// c_data_mem_type
		0,	// c_enable_rlocs
		"virtex2",	// c_family
		0,	// c_has_bfp
		1,	// c_has_bypass
		0,	// c_has_ce
		1,	// c_has_natural_output
		0,	// c_has_nfft
		0,	// c_has_ovflo
		1,	// c_has_rounding
		1,	// c_has_scaling
		0,	// c_has_sclr
		16,	 // c_input_width
		10,	 // c_nfft_max
		16,	 // c_output_width
		1,	// c_twiddle_mem_type
		16)	 // c_twiddle_width
	inst (
		.XN_RE(xn_re),
		.XN_IM(xn_im),
		.START(start),
		.FWD_INV(fwd_inv),
		.FWD_INV_WE(fwd_inv_we),
		.SCALE_SCH(scale_sch),
		.SCALE_SCH_WE(scale_sch_we),
		.CLK(clk),
		.XK_RE(xk_re),
		.XK_IM(xk_im),
		.XN_INDEX(xn_index),
		.XK_INDEX(xk_index),
		.RFD(rfd),
		.BUSY(busy),
		.DV(dv),
		.EDONE(edone),
		.DONE(done),
```

```
            .UNLOAD(),
            .NFFT(),
            .NFFT_WE(),
            .SCLR(),
            .CE(),
            .BLK_EXP(),
            .OVFLO());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of fft is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of fft is "black_box"

endmodule
//***************************************************************
`timescale 1ns/1ps

module majorfsm (clock, reset, start_pause, busy_pause, start_book, busy_book,
                 start_auto, busy_auto, freq, select, ctrl, ctrl_out, pulse, pause,
                 replay, auto, preset1, preset2, signal, button, pressed, pressed2);

  input clock, reset, busy_pause, busy_book, busy_auto, pulse, pause, replay,
       auto, preset1, preset2, select;
  output start_pause, start_book, start_auto, freq, signal, ctrl_out, pressed,
        button, pressed2;
  output [1:0] ctrl;


  reg [2:0] state, next;
  reg start_pause, start_book, start_auto, freq, signal, ctrl_out, button;
  reg [1:0] ctrl;



  parameter IDLE = 0;
  parameter CHANNEL = 1;
  parameter PB = 2;
  parameter START_AUTO = 3;
  parameter WAIT_AUTO = 4;

  assign pressed = (preset1 || preset2);
  assign pressed2 = auto;
```

```
  always @ (posedge clock) begin

    if(!reset) state <= IDLE;
    else state <= next;

       if(state == CHANNEL) ctrl <= 2'b00;
       else if(state == PB) ctrl <= 2'b01;
       else if(state == WAIT_AUTO) ctrl <= 2'b10;
       else ctrl <= ctrl;

       if(state == CHANNEL) ctrl_out <= 0;
       else if(state == PB) ctrl_out <= 1;
       else ctrl_out <= ctrl_out;

     end


  always @ (state or busy_pause or busy_book or busy_auto or pulse or auto or preset1 or preset2 or pause or replay
or select) begin

     start_pause = 0; start_book = 0; start_auto = 0; freq = 0; button = 0; signal = 0;// defaults

     case(state)

        IDLE: begin
                        if(pulse) next = CHANNEL;
            else if(auto) next = START_AUTO;
            else if(preset1 || preset2 || pause || replay) next = PB;
            else next = IDLE;
          end


        CHANNEL: begin
              next = IDLE;
              if(select) freq = 1;
              else freq = 0;
              end


        PB: begin
                    if((!busy_book) && (preset1 || preset2)) begin
           start_book = 1;
           if(preset1 == 1 && preset2 == 0) button = 0;
           else if(preset1 == 0 && preset2 == 1) button = 1;        // button tells the Minor FSMs whcih preset button
has been pressed
           else button = button;
```

```
        end
        else start_book = 0;


        if((!busy_pause) && (pause || replay)) begin
        start_pause = 1;
        if(pause == 1 && replay == 0) signal = 1;
        else if(pause == 0 && replay == 1) signal = 0;
        else signal = signal;           // signal tells minor fsm whether the request is for pause or for replay
        end
        else start_pause = 0;

        if((!busy_pause) && (!busy_book)) next = IDLE;
        else next = PB;
        end


      START_AUTO: begin
                        start_auto = 1;
            if(busy_auto) next = WAIT_AUTO;
            else next = START_AUTO;
            end

      WAIT_AUTO: begin
            if(busy_auto) next = WAIT_AUTO;
            else next = IDLE;
            end

      default: next = IDLE;
    endcase
  end
  endmodule
//**********************************************************************
  /**********************************************************************
*    This file is owned and controlled by Xilinx and must be used      *
*    solely for design, simulation, implementation and creation of     *
*    design files limited to Xilinx devices or technologies. Use       *
*    with non-Xilinx devices or technologies is expressly prohibited   *
*    and immediately terminates your license.                          *
*                                                                      *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"     *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR           *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION   *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION       *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS         *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,           *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE  *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY          *
```

```
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE                 *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR          *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF         *
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS         *
*    FOR A PARTICULAR PURPOSE.                                               *
*                                                                            *
*    Xilinx products are not intended for use in life support                *
*    appliances, devices, or systems. Use in such applications are           *
*    expressly prohibited.                                                   *
*                                                                            *
*    (c) Copyright 1995-2004 Xilinx, Inc.                                    *
*    All rights reserved.                                                    *
******************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file mult_im.v when simulating
// the core, mult_im. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Guide".

module mult_im (
      clk,
      a,
      b,
      o);    // synthesis black_box

input clk;
input [15 : 0] a;
input [15 : 0] b;
output [31 : 0] o;

// synopsys translate_off

      MULT_GEN_V7_0 #(
            9,      // bram_addr_width
            1,      // c_a_type
            16,     // c_a_width
            16,     // c_baat
            0,      // c_b_constant
            1,      // c_b_type
            "0000000000000001",     // c_b_value
            16,     // c_b_width
            0,      // c_enable_rlocs
            0,      // c_has_aclr
            0,      // c_has_a_signed
            1,      // c_has_b
```

```
            0,      // c_has_ce
            0,      // c_has_loadb
            0,      // c_has_load_done
            0,      // c_has_nd
            1,      // c_has_o
            0,      // c_has_q
            0,      // c_has_rdy
            0,      // c_has_rfd
            0,      // c_has_sclr
            0,      // c_has_swapb
            "mem",  // c_mem_init_prefix
            0,      // c_mem_type
            0,      // c_mult_type
            0,      // c_output_hold
            32,     // c_out_width
            0,      // c_pipeline
            1,      // c_reg_a_b_inputs
            0,      // c_sqm_type
            1,      // c_stack_adders
            1,      // c_standalone
            1,      // c_sync_enable
            1,      // c_use_luts
            0)      // c_v2_speed
        inst (
            .CLK(clk),
            .A(a),
            .B(b),
            .O(o),
            .Q(),
            .A_SIGNED(),
            .LOADB(),
            .LOAD_DONE(),
            .SWAPB(),
            .CE(),
            .ACLR(),
            .SCLR(),
            .RFD(),
            .ND(),
            .RDY());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of mult_im is "true"

// XST black box declaration
```

```
// box_type "black_box"
// synthesis attribute box_type of mult_im is "black_box"

endmodule

//*****************************************************************
/*****************************************************************************
*    This file is owned and controlled by Xilinx and must be used          *
*    solely for design, simulation, implementation and creation of         *
*    design files limited to Xilinx devices or technologies. Use           *
*    with non-Xilinx devices or technologies is expressly prohibited       *
*    and immediately terminates your license.                              *
*                                                                          *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"         *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR               *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION       *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION           *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS             *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,               *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE      *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY              *
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE               *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR        *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF       *
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS       *
*    FOR A PARTICULAR PURPOSE.                                             *
*                                                                          *
*    Xilinx products are not intended for use in life support              *
*    appliances, devices, or systems. Use in such applications are         *
*    expressly prohibited.                                                 *
*                                                                          *
*    (c) Copyright 1995-2004 Xilinx, Inc.                                  *
*    All rights reserved.                                                  *
*****************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file mult_re.v when simulating
// the core, mult_re. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Guide".

module mult_re (
        clk,
        a,
        b,
        o);    // synthesis black_box
```

```
input clk;
input [15 : 0] a;
input [15 : 0] b;
output [31 : 0] o;

// synopsys translate_off

        MULT_GEN_V7_0 #(
                9,      // bram_addr_width
                1,      // c_a_type
                16,     // c_a_width
                16,     // c_baat
                0,      // c_b_constant
                1,      // c_b_type
                "0000000000000001",     // c_b_value
                16,     // c_b_width
                0,      // c_enable_rlocs
                0,      // c_has_aclr
                0,      // c_has_a_signed
                1,      // c_has_b
                0,      // c_has_ce
                0,      // c_has_loadb
                0,      // c_has_load_done
                0,      // c_has_nd
                1,      // c_has_o
                0,      // c_has_q
                0,      // c_has_rdy
                0,      // c_has_rfd
                0,      // c_has_sclr
                0,      // c_has_swapb
                "mem",  // c_mem_init_prefix
                0,      // c_mem_type
                0,      // c_mult_type
                0,      // c_output_hold
                32,     // c_out_width
                0,      // c_pipeline
                1,      // c_reg_a_b_inputs
                0,      // c_sqm_type
                1,      // c_stack_adders
                1,      // c_standalone
                1,      // c_sync_enable
                1,      // c_use_luts
                0)      // c_v2_speed
        inst (
                .CLK(clk),
                .A(a),
                .B(b),
```

```
            .O(o),
            .Q(),
            .A_SIGNED(),
            .LOADB(),
            .LOAD_DONE(),
            .SWAPB(),
            .CE(),
            .ACLR(),
            .SCLR(),
            .RFD(),
            .ND(),
            .RDY());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of mult_re is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of mult_re is "black_box"

endmodule

//**************************************************************
  module pausefsm(clk, reset, start_pause, signal, busy_pause, WE, address);

   input clk, reset, start_pause, signal;
   output busy_pause, WE;
   output [16:0] address;

   reg busy_pause, WE;
   reg [10:0] counter;
   reg [16:0] address;
   reg [1:0] state, next;

   parameter IDLE = 0;
   parameter PAUSE = 1;
   parameter REPLAY = 2;

   always @ (posedge clk) begin
     if(!reset) state <= IDLE;
     else state <= next;

     if (state == IDLE) begin
            counter <= 11'd0;
```

```
            address <= 17'd0;
            end
    else if((state == PAUSE || state == REPLAY) && counter == 11'd1146) begin
      counter <= 11'd0;
      address <= address + 1;
    end
    else if((state == PAUSE || state == REPLAY) && counter != 11'd1146)begin
      address <= address;
      counter <= counter + 1;
    end
    else begin
       address <= address;
       counter <= counter;
    end
  end

   always @ (state or start_pause) begin

     WE = 0; busy_pause = 1;// default

     case(state)

        IDLE: begin
              busy_pause = 0;
              if(start_pause == 1 && signal == 1) next = PAUSE;
              else if(start_pause == 1 && signal == 0) next = REPLAY;
              else next = IDLE;
             end


        PAUSE: begin
                busy_pause = 1;
                WE = 1;
                if(address == 17'd131071) next = IDLE;
                else next = PAUSE;
               end


        REPLAY: begin
                busy_pause = 1;
                if(address == 17'd131071) next = IDLE;
                else next = REPLAY;
              end
         default: next = IDLE;

    endcase
end
endmodule
```

```
//*****************************************************
module synchro(clk, reset, frequency, pause_switch, replay_switch, preset1_switch, preset2_switch, auto_switch,
        Reset_sync, freq_sync, pause_sync, replay_sync, preset1_sync, preset2_sync, auto_sync);

input clk, reset, frequency, pause_switch, replay_switch, preset1_switch, preset2_switch, auto_switch;
output Reset_sync, freq_sync, pause_sync, replay_sync, preset1_sync, preset2_sync, auto_sync;

reg f1, p1, rep1, pre1, pre2, a1, r1, Reset_sync, freq_sync, pause_sync, replay_sync, preset1_sync,
       preset2_sync, auto_sync;

always @ (posedge clk)
begin
r1 <= reset;
Reset_sync <= r1;

f1 <= frequency;
freq_sync <= f1;

p1 <= pause_switch;
pause_sync <= p1;

rep1 <= replay_switch;
replay_sync <= rep1;

pre1 <= preset1_switch;
preset1_sync <= pre1;

pre2 <= preset2_switch;
preset2_sync <= pre2;

a1 <= auto_switch;
auto_sync <= a1;
end
endmodule
//*************************************************
`timescale 1ns/1ps
module top(clk, reset, frequency, pause, replay, preset1, preset2, auto, scan,
         freq_dsp, v_sync, busy_write, busy_display, adc_in, /*test1, test2, dout1, dout2,
             button_left, button_right,*/ RAMaddress, data_fft,sum);



   input clk, reset, frequency, pause, replay, preset1, preset2, auto, scan, v_sync,
        busy_display; /*test1, test2, button_left, button_right*/
   input [13:0] adc_in;
   input [9:0] RAMaddress;
   output freq_dsp, busy_write;
   output [15:0] data_fft;
```

```
output [16:0] sum;
reg [9:0] ind, actual;

wire Reset_Sync, enable, start_timer, reset_timer, expired, freq_sync, pause_sync,
    replay_sync, preset1_sync, preset2_sync, auto_sync, pulse, start_pause,
    busy_pause, start_book, busy_book, start_auto, busy_auto, freq, ctrl_out,
    signal, pressed, button, WE_book, book_address, freq_out, WE_pause, pressed2,
    start, scale_sch_we, fwd_inv, fwd_inv_we, fft_done, freq3, rfd,
    busy, dv, edone, WE_fft;

wire [1:0] ctrl;
wire [9:0] scale_sch;
wire [3:0] data_in, data_w;
wire [9:0] xn_index, xk_index, actual2;
wire [16:0] pause_address, sum;
wire [15:0] xk_re, xk_im;
wire [31:0] re_sq, im_sq;



timer timer1(.clk(clk), .Reset_Sync(reset_timer), .enable(enable),
          .start_timer(start_timer), .expired(expired));

divider divider1(.clk(clk), .Reset_Sync(Reset_Sync), .enable(enable));

synchro synchronizer(.clk(clk), .reset(reset), .frequency(frequency),
                .pause_switch(pause), .replay_switch(replay),
                .preset1_switch(preset1), .preset2_switch(preset2),
                .auto_switch(auto), .Reset_sync(Reset_Sync),
                .freq_sync(freq_sync), .pause_sync(pause_sync),
                .replay_sync(replay_sync), .preset1_sync(preset1_sync),
                .preset2_sync(preset2_sync), .auto_sync(auto_sync));


level_to_pulse level_to_pulse1(.clk(clk), .freq(freq_sync), .pulse(pulse));

majorfsm majorfsm1(.clock(clk), .reset(Reset_Sync), .start_pause(start_pause),
                .busy_pause(busy_pause), .start_book(start_book),
                .busy_book(busy_book), .start_auto(start_auto),
                .busy_auto(busy_auto), .freq(freq), .select(frequency),
                .ctrl(ctrl), .ctrl_out(ctrl_out), .pulse(pulse),
                .pause(pause_sync), .replay(replay_sync), .auto(auto_sync),
                .preset1(preset1_sync), .preset2(preset2_sync), .signal(signal),
                .button(button), .pressed(pressed), .pressed2(pressed2));



bookfsm bookfsm1(.clk(clk), .reset(Reset_Sync), .expired(expired), .button(button),
```

```
            .pressed(pressed), .start_book(start_book), .busy_book(busy_book),
            .reset_timer(reset_timer), .WE_book(WE_book), .data_w(data_w),
            .data_in(data_in), .address(book_address), .freq_out(freq_out));

 bookram bookram1 (.addr(book_address), .clk(clk), .din(data_w), .dout(data_in), .we(WE_book));


 pausefsm pausefsm1(.clk(clk), .reset(Reset_Sync), .start_pause(start_pause),
              .signal(signal), .busy_pause(busy_pause), .WE(WE_pause),
             .address(pause_address));


 autofsm autofsm1(.clk(clk), .reset(Reset_Sync), .start_auto(start_auto),
            .busy_auto(busy_auto), .v_sync(v_sync), .start(start),
            .scale_sch(scale_sch), .scale_sch_we(scale_sch_we),
            .fwd_inv(fwd_inv), .fwd_inv_we(fwd_inv_we), .fft_done(fft_done),
            .busy_write(busy_write), .busy_display(busy_display), .scan(scan),
            .freq3(freq3), .data_in(data_fft), .pressed2(pressed2),
            .WE_fft(WE_fft));




fft fft1(.xn_re({adc_in, 2'b00}),     .xn_im(16'd0), .start(start), .fwd_inv(fwd_inv),
          .fwd_inv_we(fwd_inv_we),        .scale_sch(scale_sch),
          .scale_sch_we(scale_sch_we),    .clk(clk),      .xk_re(xk_re),  .xk_im(xk_im),
          .xn_index(xn_index),    .xk_index(xk_index),    .rfd(rfd),      .busy(busy),    .dv(dv),
               .edone(edone),  .done(fft_done));


   assign actual2 = (!v_sync) ? actual : RAMaddress;

     fftram fftram1(.addr(actual2), .clk(clk),       .din(sum[16:1]),        .dout(data_fft),
                .we(WE_fft));

     mult_re multx(.clk(clk),.a(xk_re),.b(xk_re),.o(re_sq));

     mult_im multy(.clk(clk),        .a(xk_im),      .b(xk_im),      .o(im_sq));

     adder adder1(.A(re_sq[15:0]),   .B(im_sq[15:0]),        .Q(sum),        .CLK(clk));



assign freq_dsp = ctrl[1] ? (ctrl[0] ? freq_dsp : freq3) : (ctrl[0] ? freq_out : freq);

always @ (posedge clk) begin
   ind <= xk_index;
   actual <= ind;
```

```
end

endmodule
```