DE2 Board Amendment

Last updated 8/25/2012

Introduction

The HDL codes and C codes in the book are developed for the Altera DE1 board. Since the DE2 board has similar I/O configurations. Most codes can be directly used for the DE2 board. Only the VGA controller HDL codes require minor modifications. The following sections discuss the modified HDL codes and Nios II testing system.

VGA IP cores

While the DE1 board uses a DAC resister network that supports only 12-bit color (i.e., 4 bits each channel), the DE2 board uses a video DAC chip that supports 30-bit color (i.e., 10 bits each channel). In addition to the 30 color signals, the DAC chip includes several additional signals:

- vdac_clk: a clock signal for the chip, the 30 color signals are sampled at the rising edge of this clock. The pixel clock signal, p_tick, of the VGA synchronization circuit can be used for this purpose.
- vdac_blank_n: an active-low control signal that blanks the screen when asserted. The video_on_reg signal of the VGA synchronization circuit can be used for this purpose.
- vdac_sync_n: a special active-low synchronization signal. It is not used in our designed and can be tied to '0'.

To accommodate the new chip, the HDL codes of the palette circuit (Listing 18.3) and the top-level Avalon wrapping circuit (Listing 18.4) needs to be revised. The revised codes are listed in the appendix.

A new VGA IP core, chu_avalon_vga_de2, is constructed based on the new circuits and should be used to replace the original chu_avalon_vga core. The procedure to use the new core is

- Copy the chu_avalon_vga_de2 directory and its files to the location where other IP cores are stored.
- Follow the procedure in Section 16.10.1 to integrate the new IP core into SOPC Builder.
- Use the chu_avalon_vga_de2 core in place of chu_avalon_vga.

Comprehensive Nios II system

A comprehensive Nios II system that contains all main IP cores in Parts III and IV is constructed for the DE1 board in Section 17.10. A similar system can also be constructed for the DE2 board. Several revisions are needed:

- Use the new VGA IP core, chu_avalon_vga_de2, for the VGA controller.
- Extend the number of sliding switches from 10 to 18.
- Extend the number of discrete red LEDs from 10 to 18.
- Extend the number of seven-segment displays from 4 to 8.

The revised top-level code (Listing 17.14) is shown in the appendix. To maintain the software compatibility, the additional LEDs and switches are not used. They can be easily enabled by modifying the corresponding PIO cores and C codes.

Pin assignment file

The DE2 board requires different pin assignment. A new csv file, chu_de2_pin.csv, is included to replace the chu_de1_pin.csv file discussed in Section 3.5.1.

Modified files

The modified files are included in the chu_ip_vhdl_de2 directory:

- chu_avalon_vga_de2: directory containing the files for the new VGA core.
- chu_de2_pin.csv: new pin assignment file to replace chu_de1_pin.csv.
- list_17_14_p34_top_de2.v: new file to replace original Listing 17.4.
- list_18_03_palette_de2.v: new file to replace original Listing 18.3.
- list_18_04_chu_avalon_vga_de2.v: new file to replace original Listing 18.4.
- nios_p34_de2.sopc: the new .sopc file used in conjunction with Listing 17.4.

Appendix 1: Listing 18.3

```
module palette_de2
   (
    input wire [7:0] color_in,
    output wire [29:0] color_out
   );

   wire [9:0] r10, g10, b10;
   // body
   // 3-bit red to 10-bit red, 3-bit green to 10-bit green
   // 2-bit blue to 10-bit blue
   assign r10 = {{3{color_in[7:5]}}, color_in[7]};
   assign g10 = {{3{color_in[4:2]}}, color_in[4]};
   assign b10 = {5{color_in[1:0]}};
   assign color_out = {r10, g10, b10};
endmodule
```

Appendix 2: Listing 18.4

```
module chu_avalon_vga_de2
   (
    input wire clk, reset,
    // Avalon MM interface
    input wire [19:0] vga_address,
    input wire vga_chipselect, vga_write, vga_read,
    input wire [31:0] vga_writedata,
    output wire [31:0] vga_readdata,
    // conduit (to VGA monitor)
    output wire vsync, hsync,
    //**************** for DE2 board ******************
    output wire [29:0] rgb,
    output wire vdac_clk,
    output wire vdac_blank_n,
    output wire vdac_sync_n,
    //*************************************************
    // conduit (to/from SRAM)
    output wire [17:0] sram_addr,
    inout [15:0] sram_dq,
    output sram_ce_n, sram_oe_n, sram_we_n,
    output sram_lb_n, sram_ub_n
 );

   // signal declaration
   reg video_on_reg, vsync_reg, hsync_reg;
   wire vsync_i, hsync_i, video_on_i, p_tick;
   wire [9:0] pixel_x, pixel_y;
   wire wr_vram, rd_vram;
   wire [7:0] cpu_rd_data, vga_rd_data;
   wire [29:0] color;

   // body
   //===================================================================
   // instantiation
   //===================================================================
   // instantiate VGA sync circuit
   vga_sync sync_unit
      (.clk(clk), .reset(reset), .hsync_i(hsync_i), .vsync_i(vsync_i),
       .video_on_i(video_on_i), .p_tick(p_tick),
       .pixel_x(pixel_x), .pixel_y(pixel_y));
   // instantiate video SRAM control
   vram_ctrl vram_unit
      (.clk(clk), .reset(reset),
       // from video sync
       .pixel_x(pixel_x), .pixel_y(pixel_y), .p_tick(p_tick),
       // avalon bus interface
       .vga_rd_data(vga_rd_data), .cpu_rd_data(cpu_rd_data),
       .cpu_wr_data(vga_writedata[7:0]),
       .cpu_addr(vga_address[18:0]),
       .cpu_mem_wr(wr_vram), .cpu_mem_rd(rd_vram),
       // to/from SRAM chip
       .sram_addr(sram_addr), .sram_dq(sram_dq),
       .sram_we_n(sram_we_n), .sram_oe_n(sram_oe_n), .sram_ce_n(sram_ce_n),
       .sram_ub_n(sram_ub_n), .sram_lb_n(sram_lb_n)
      );
   // instantiate palette table (8-bit to 12-bit conversion)
   palette_de2 palet_unit
      (.color_in(vga_rd_data), .color_out(color));
   //===================================================================
   // registers, write decoding, and read multiplexing
   //===================================================================
   // delay vga sync to accomodate memory access
   always @(posedge clk)
      if (p_tick)
         begin
            vsync_reg <= vsync_i;
            hsync_reg <= hsync_i;
            video_on_reg <= video_on_i;
         end
   assign vsync = vsync_reg;
   assign hsync = hsync_reg;
   // memory read/write decoding
   assign wr_vram = vga_write & vga_chipselect &  ~vga_address[19];
   assign rd_vram = vga_read & vga_chipselect &  ~vga_address[19];
   // read data mux
   assign vga_readdata = ~vga_address[19] ? {24'b0, cpu_rd_data} :
                                            {12'b0, pixel_y, pixel_x};
   // video output
   assign rgb = video_on_reg ? color : 12'b0;
   //***************   For DE2  ****************************************
   assign vdac_sync_n = 0;              // lower voltage in blank period
   assign vdac_blank_n = video_on_reg;
   assign vdac_clk = p_tick;            // clock for the video DAC
   //******************************************************************

endmodule
```

Appendix 3: Listing 17.14

```
module top_p34_de2
   (
    input wire clk,
    // switch and LED
    input wire [9:0] sw,
    input wire [3:0] key,
    output wire [7:0] ledg,
    // ************** for DE2 ***************************************
    output wire [17:0] ledr,
    output wire [6:0] hex7, hex6, hex5, hex4, hex3, hex2, hex1, hex0,
    // **************************************************************
    // PS2
    inout ps2c, ps2d,
    // VGA
    output wire vsync, hsync,
    // ************** for DE2 ***************************************
    output wire [29:0] rgb,
    output wire vdac_clk,       // used by DE2 dac
    output wire vdac_blank_n,   // used by DE2 dac
    output wire vdac_sync_n,    // used by DE2 dac
    // **************************************************************
    // audio codec
    output wire m_clk, b_clk, dac_lr_clk, adc_lr_clk,
    output wire dacdat,
    input wire adcdat,
    output wire i2c_sclk,
    inout wire i2c_sdat,
    // SD card
    output wire sd_clk, sd_di, sd_cs,
    input wire  sd_do,
    // SRAM
    output wire [17:0] sram_addr,
    inout wire [15:0] sram_dq,
    output wire sram_ce_n, sram_oe_n, sram_we_n,
    output wire sram_lb_n, sram_ub_n,
    // SDRAM
    output wire dram_clk,
    output wire dram_cs_n, dram_cke, dram_ldqm, dram_udqm,
    output wire dram_cas_n, dram_ras_n, dram_we_n,
    output wire [11:0] dram_addr,
    output wire dram_ba_0, dram_ba_1,
    inout wire [15:0] dram_dq
   );

   // signal declaration
   wire [18:0] led;
   wire [31:0] sseg;
   wire [15:0] ddfs_data;
   wire dac_load_tick;

   // body
   // instantiate nios
   nios_p34_de2 cpu_unit
     (.clk_50M(clk),
      .clk_sys(),
      .clk_sdram(dram_clk),
      .reset_n(key[3]),
      // switch and LED
      .in_port_to_the_btn({1'b0,key[2:0]}),
      .in_port_to_the_switch(sw),
      .out_port_from_the_led(led),
      .out_port_from_the_sseg(sseg),
      // PS2
      .ps2c_to_and_from_the_ps2(ps2c),
      .ps2d_to_and_from_the_ps2(ps2d),
      //  VGA monitor
      // ************** for DE2 ***********************
      .vdac_clk_from_the_vram(vdac_clk),
      .vdac_sync_n_from_the_vram(vdac_sync_n),
      .vdac_blank_n_from_the_vram(vdac_blank_n),
      // **********************************************
      .hsync_from_the_vram(hsync),
      .vsync_from_the_vram(vsync),
      .rgb_from_the_vram(rgb),
      // VGA SRAM
      .sram_addr_from_the_vram(sram_addr),
      .sram_ce_n_from_the_vram(sram_ce_n),
      .sram_dq_to_and_from_the_vram(sram_dq),
      .sram_lb_n_from_the_vram(sram_lb_n),
      .sram_oe_n_from_the_vram(sram_oe_n),
      .sram_ub_n_from_the_vram(sram_ub_n),
      .sram_we_n_from_the_vram(sram_we_n),
      // audio codec
      .b_clk_from_the_audio(b_clk),
      .m_clk_from_the_audio(m_clk),
      .dac_lr_clk_from_the_audio(dac_lr_clk),
      .adc_lr_clk_from_the_audio(adc_lr_clk),
      .dacdat_from_the_audio(dacdat),
      .adcdat_to_the_audio(adcdat),
      .i2c_sclk_from_the_audio(i2c_sclk),
      .i2c_sdat_to_and_from_the_audio(i2c_sdat),
      .codec_adc_data_out_from_the_audio(),
      .codec_adc_rd_to_the_audio(dac_load_tick),
      .codec_dac_data_in_to_the_audio({ddfs_data, ddfs_data}),
      .codec_dac_wr_to_the_audio(dac_load_tick),
      .codec_sample_tick_from_the_audio(dac_load_tick),
      // SD card
      .sd_clk_from_the_sdc(sd_clk),
      .sd_do_to_the_sdc(sd_do),
      .sd_di_from_the_sdc(sd_di),
      .sd_cs_from_the_sdc(sd_cs),
      // DDFS
      .ddfs_data_out_from_the_d_engine(ddfs_data),
       // SDRAM
      .zs_addr_from_the_sdram(dram_addr),
      .zs_ba_from_the_sdram({dram_ba_1, dram_ba_0}),
      .zs_cas_n_from_the_sdram(dram_cas_n),
      .zs_cke_from_the_sdram(dram_cke),
      .zs_cs_n_from_the_sdram(dram_cs_n),
      .zs_dq_to_and_from_the_sdram(dram_dq),
      .zs_dqm_from_the_sdram({dram_udqm, dram_ldqm}),
      .zs_ras_n_from_the_sdram(dram_ras_n),
      .zs_we_n_from_the_sdram(dram_we_n)
    );
    // output assignment
    assign hex3 = sseg[30:24];
    assign hex2 = sseg[22:16];
    assign hex1 = sseg[14:8];
    assign hex0 = sseg[6:0];
    assign ledr[9:0] = led[17:8];
    assign ledg = led[7:0];
    // ************* for DE2 ***********************
    assign ledr[17:10] = 8'h00;
```

```
    assign hex7 = 7'b1111111;
    assign hex6 = 7'b1111111;
    assign hex5 = 7'b1111111;
    assign hex4 = 7'b1111111;
    // ***********************************************
endmodule
```