DE2 Board Amendment
Last updated 10/10/2011

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 17.3) and the top-level Avalon wrapping circuit (Listing 17.4) 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 16.10. 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 16.14) 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_16_14_p34_top_de2.vhd: new file to replace original Listing 16.4.
- list_17_03_palette_de2.vhd: new file to replace original Listing 17.3.
- list_17_04_ chu_avalon_vga_de2.vhd: new file to replace original Listing 17.4.
- nios_p34_de2.sopc: the new .sopc file used in conjunction with Listing 16.4.

Appendix 1: Listing 17.3

```
library ieee;
use ieee.std_logic_1164.all;
entity palette_de2 is
   port (
      color_in: in  std_logic_vector(7 downto 0);
      color_out: out std_logic_vector(29 downto 0)
   );
end palette_de2;

architecture arch of palette_de2 is
   signal r3, g3:  std_logic_vector(2 downto 0);
   signal b2:  std_logic_vector(1 downto 0);
begin
   r3 <= color_in(7 downto 5);          -- 3-bit red
   g3 <= color_in(4 downto 2);          -- 3-bit green
   b2 <= color_in(1 downto 0);          -- 2-bit blue
   color_out <= r3 & r3 & r3 & r3(2) &  -- 10-bit red
                g3 & g3 & g3 & g3(2) &  -- 10-bit green
                b2 & b2 &b2 &b2 & b2;   -- 10-bit blue
end arch;
```

Appendix 2: Listing 17.4

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity chu_avalon_vga_de2 is
   port (
      clk, reset: in  std_logic;
      -- to vga monitor
      hsync, vsync: out std_logic;
      -- ***************   FOR DE2  ******************
      rgb: out std_logic_vector(29 downto 0); -- DE2
      vdac_clk: out std_logic;     -- used by DE2 dac
      vdac_blank_n: out std_logic; -- used by DE2 dac
      vdac_sync_n: out std_logic;  -- used by DE2 dac
      -- ******************************************
      -- to/from SRAM chip
      sram_addr: out std_logic_vector(17 downto 0);
      sram_dq: inout std_logic_vector(15 downto 0);
      sram_we_n, sram_oe_n: out std_logic;
      sram_ce_n, sram_ub_n, sram_lb_n: out std_logic;
      -- avalon interface
      vga_address: in  std_logic_vector(19 downto 0);
      vga_chipselect: in  std_logic;
      vga_write: in std_logic;
      vga_read: in std_logic;
      vga_writedata: in std_logic_vector(31 downto 0);
      vga_readdata: out std_logic_vector(31 downto 0)
 );
end chu_avalon_vga_de2;

architecture arch of chu_avalon_vga_de2 is
   signal video_on_reg,video_on_i: std_logic;
   signal vsync_i, hsync_i: std_logic;
   signal pixel_x, pixel_y: std_logic_vector(9 downto 0);
   signal p_tick: std_logic;
   signal cpu_rd_data: std_logic_vector(7 downto 0);
   signal vga_rd_data: std_logic_vector(7 downto 0);
   signal color: std_logic_vector(29 downto 0);
   signal wr_vram, rd_vram: std_logic;
begin
   -- instantiate VGA sync circuit
   vga_sync_unit: entity work.vga_sync
      port map(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_unit: entity work.vram_ctrl
   port map (
      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 downto 0),
      cpu_addr=>vga_address(18 downto 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)
   palet_unit: entity work.palette_de2
   port map (
      color_in=>vga_rd_data,
      color_out=>color);
   -- delay vga sync to accomodate memory access
   process (clk)
   begin
      if (clk'event and clk='1') then
         if (p_tick='1') then
            vsync <= vsync_i;
            hsync <= hsync_i;
            video_on_reg <= video_on_i;
         end if;
      end if;
   end process;
   -- memory read/write decoding
   wr_vram <=
      '1' when vga_write='1' and vga_chipselect='1' and
               vga_address(19)='0' else
      '0';
   rd_vram <=
      '1' when vga_read='1' and vga_chipselect='1' and
               vga_address(19)='0' else
      '0';
   -- input data mux
   vga_readdata <= x"000000" & cpu_rd_data when vga_address(19)='0' else
                   x"000" & pixel_y & pixel_x;
   rgb <= (others=>'0') when video_on_reg='0' else color;
   -- ***************   FOR DE2  ****************************
   vdac_sync_n <= '0';   -- lower volatge in blank period
   vdac_blank_n <= video_on_reg;
   vdac_clk <= p_tick;   -- clock for the video DAC
   -- ******************************************************

end arch;
```

Appendix 3: Listing 16.14

```
library ieee;
use ieee.std_logic_1164.all;
entity p34_top_de2 is
   port(
      clk: in std_logic;
      -- switch/LEDs
      sw: in std_logic_vector(9 downto 0);
      key: in std_logic_vector(3 downto 0);
      ledg: out std_logic_vector(7 downto 0);
      -- ***************   FOR DE2  ******************
      hex7, hex6, hex5, hex4, hex3, hex2, hex1, hex0:
         out std_logic_vector(6 downto 0);
      ledr: out std_logic_vector(17 downto 0);
      -- *****************************************
      -- to/from ps2
      ps2c, ps2d: inout std_logic;
      -- to VGA
      vsync, hsync: out std_logic;
      -- ***************   FOR DE2  ******************
      rgb: out std_logic_vector(29 downto 0); -- DE2
      vdac_clk: out std_logic;     -- used by DE2 dac
      vdac_blank_n: out std_logic; -- used by DE2 dac
      vdac_sync_n: out std_logic;  -- used by DE2 dac
      -- *****************************************
      -- to/from audiocodec
      m_clk, b_clk, dac_lr_clk, adc_lr_clk: out std_logic;
      dacdat: out std_logic;
      adcdat: in std_logic;
      i2c_sclk: out std_logic;
      i2c_sdat: inout std_logic;
      -- to/from SD card
      sd_clk: out std_logic;
      sd_di: out std_logic;
      sd_do: in std_logic;
      sd_cs: out std_logic;
      -- to/from SRAM
      sram_addr: out std_logic_vector (17 downto 0);
      sram_dq: inout std_logic_vector (15 downto 0);
      sram_ce_n: out std_logic;
      sram_lb_n: out std_logic;
      sram_oe_n: out std_logic;
      sram_ub_n: out std_logic;
      sram_we_n: out std_logic;
      -- to/from SDRAM
      dram_clk: out std_logic;
      dram_cs_n, dram_cke: out std_logic;
      dram_ldqm, dram_udqm: out std_logic;
      dram_cas_n, dram_ras_n, dram_we_n: out std_logic;
      dram_addr: out std_logic_vector(11 downto 0);
      dram_ba_0, dram_ba_1: out std_logic;
      dram_dq: inout std_logic_vector(15 downto 0)
   );
end p34_top_de2;

architecture structure of p34_top_de2 is
   component nios_p34_de2 is
   port (
      -- clock and reset
      signal clk_50m: in std_logic;
      signal clk_sdram: out std_logic;
```

```
signal clk_sys: out std_logic;
signal reset_n: in std_logic;
-- SDRAM
signal zs_addr_from_the_sdram:
       out std_logic_vector (11 downto 0);
signal zs_ba_from_the_sdram:
       out std_logic_vector (1 downto 0);
signal zs_dq_to_and_from_the_sdram:
       inout std_logic_vector (15 downto 0);
signal zs_cas_n_from_the_sdram: out std_logic;
signal zs_cke_from_the_sdram: out std_logic;
signal zs_cs_n_from_the_sdram: out std_logic;
signal zs_ras_n_from_the_sdram: out std_logic;
signal zs_we_n_from_the_sdram: out std_logic;
signal zs_dqm_from_the_sdram:
       out std_logic_vector (1 downto 0);
-- siwtch and leds
signal in_port_to_the_switch:
       in std_logic_vector (9 downto 0);
signal in_port_to_the_btn:
       in std_logic_vector (3 downto 0);
signal out_port_from_the_led:
       out std_logic_vector (17 downto 0);
signal out_port_from_the_sseg:
       out std_logic_vector (31 downto 0);
-- PS2
signal ps2c_to_and_from_the_ps2: inout std_logic;
signal ps2d_to_and_from_the_ps2: inout std_logic;
-- video RAM (VGA)
signal hsync_from_the_vram: out std_logic;
signal vsync_from_the_vram: out std_logic;
-- ***************   FOR DE2  ******************
signal rgb_from_the_vram:
       out std_logic_vector (29 downto 0);
signal vdac_clk_from_the_vram: out std_logic;
signal vdac_blank_n_from_the_vram: out std_logic;
signal vdac_sync_n_from_the_vram: out std_logic;
-- ******************************************
-- video RAM (SRAM)
signal sram_addr_from_the_vram:
       out std_logic_vector (17 downto 0);
signal sram_dq_to_and_from_the_vram:
       inout std_logic_vector (15 downto 0);
signal sram_ce_n_from_the_vram:out std_logic;
signal sram_lb_n_from_the_vram: out std_logic;
signal sram_oe_n_from_the_vram: out std_logic;
signal sram_ub_n_from_the_vram: out std_logic;
signal sram_we_n_from_the_vram: out std_logic;
-- audio codec
signal m_clk_from_the_audio: out std_logic;
signal b_clk_from_the_audio: out std_logic;
signal adc_lr_clk_from_the_audio: out std_logic;
signal adcdat_to_the_audio: in std_logic;
signal dac_lr_clk_from_the_audio: out std_logic;
signal dacdat_from_the_audio: out std_logic;
signal codec_adc_data_out_from_the_audio:
       out std_logic_vector (31 downto 0);
signal codec_dac_data_in_to_the_audio:
       in std_logic_vector (31 downto 0);
signal codec_adc_rd_to_the_audio: in std_logic;
signal codec_dac_wr_to_the_audio: in std_logic;
signal codec_sample_tick_from_the_audio: out std_logic;
signal i2c_sclk_from_the_audio: out std_logic;
```

```
      signal i2c_sdat_to_and_from_the_audio: inout std_logic;
      -- SD card
      signal sd_clk_from_the_sdc: out std_logic;
      signal sd_cs_from_the_sdc: out std_logic;
      signal sd_di_from_the_sdc: out std_logic;
      signal sd_do_to_the_sdc: in std_logic;
      -- DDFS
      signal ddfs_data_out_from_the_d_engine:
             out std_logic_vector (15 downto 0)
      );
   end component nios_p34_de2;
   signal sseg4: std_logic_vector(31 downto 0);
   signal led: std_logic_vector(17 downto 0);
   signal ddfs_data: std_logic_vector(15 downto 0);
   signal dac_load_tick: std_logic;

begin
   nios: nios_p34_de2
   port map(
      clk_50M=>clk,
      clk_sdram=>dram_clk,
      clk_sys=>open,
      reset_n=>key(3),  --*****
      in_port_to_the_btn=> '0'&key(2 downto 0),
      in_port_to_the_switch=>sw,
      out_port_from_the_led=>led,
      out_port_from_the_sseg=>sseg4,
      -- SDRAM
      zs_addr_from_the_sdram=>dram_addr,
      zs_ba_from_the_sdram(1)=>dram_ba_1,
      zs_ba_from_the_sdram(0)=>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_ras_n_from_the_sdram=>dram_ras_n,
      zs_we_n_from_the_sdram=>dram_we_n,
      zs_dqm_from_the_sdram(1)=> dram_udqm,
      zs_dqm_from_the_sdram(0)=> dram_ldqm,
      -- PS2
      ps2c_to_and_from_the_ps2=>ps2c,
      ps2d_to_and_from_the_ps2=>ps2d,
      -- video ram
      -- ***************   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,
      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=>open,
      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
    );
    ledr(9 downto 0) <= led(17 downto 8);
    ledg <= led(7 downto 0);
    hex3 <= sseg4(30 downto 24);
    hex2 <= sseg4(22 downto 16);
    hex1 <= sseg4(14 downto 8);
    hex0 <= sseg4(6 downto 0);
    -- ***************   FOR DE2  ******************
    ledr(17 downto 10) <= (others=>'0');
    hex7 <= (others=>'1');
    hex6 <= (others=>'1');
    hex5 <= (others=>'1');
    hex4 <= (others=>'1');
end structure;
```