```
  1: ----------------------------------------------------------------------------------
  2: -- Refer to cpt_ovm_bram.schdoc for diagram
  3: -- Chris Brown / David Bell
  4: --
  5: -- Uses VGA signals from camera to put pixel data in BRAM, maintaining line/frame
  6: -- count for use at MUX stage (later).
  7: -- Allows reads from BRAM in internal clock domain, prechecked by the desired
  8: -- burst length.
  9: --
 10: ----------------------------------------------------------------------------------
 11: library IEEE;
 12: use IEEE.STD_LOGIC_1164.ALL;
 13: use ieee.numeric_std.all;
 14:
 15: -- Uncomment the following library declaration if instantiating
 16: -- any Xilinx primitives in this code.
 17: library UNISIM;
 18: use UNISIM.VComponents.all;
 19:
 20:
 21: library util;
 22: use util.pkg_util.all;
 23:
 24:
 25: entity cpt_ovm_bram is
 26:         port(
 27:                 i_pclk : in std_logic;
 28:                 i_vsync : in std_logic;
 29:                 i_href : in std_logic;
 30:                 i_data : in std_logic_vector (7 downto 0);
 31:                 i_reset : in std_logic;
 32:
 33:                 o_rd_data : out std_logic_vector(31 downto 0);
 34:                 o_frame_number : out integer range 0 to 3;
 35:                 o_line_number : out integer range 0 to 2047;
 36:
 37:                 o_words_read: out integer range 0 to 511;
 38:                 i_burst_length : std_logic_vector(5 downto 0);
 39:                 o_burst_available : out std_logic;
 40:                 o_collision : out std_logic;
 41:
 42:                 i_clk : in std_logic;
 43:                 i_rd_enable : in std_logic
 44:                 );
 45:
 46: end cpt_ovm_bram;
 47:
 48: architecture Behavioral of cpt_ovm_bram is
 49:
 50:         COMPONENT cpt_upcounter
 51:         generic (
 52:                 INIT : integer := -1
 53:         );
 54:         PORT(
 55:                 i_clk : IN std_logic;
 56:                 i_enable : IN std_logic;
 57:                 i_lowest : IN integer;
 58:                 i_highest : IN integer;
 59:                 i_increment : IN integer;
 60:                 i_clear : IN std_logic;
 61:                 i_preset : IN std_logic;
 62:                 o_count : OUT integer;
 63:                 o_carry : OUT std_logic
 64:                 );
 65:         END COMPONENT;
 66:
 67:         signal data : std_logic_vector(31 downto 0);
 68:
 69:         signal pclk : std_logic;
 70:         signal pclk_n : std_logic;
 71:
 72:         signal vsync : std_logic;
 73:         signal vsync_d1 : std_logic;
 74:
 75:         signal frame_count_en : std_logic;
 76:
 77:         signal href : std_logic;
 78:         signal href_d1 : std_logic;
 79:         signal line_count_en : std_logic;
 80:
 81:         signal bytes_written : integer range 0 to 2**12-1;
 82:         signal words_written : integer range 0 to 2**10-1;
 83:
 84:         signal pixel_addr : std_logic_vector (13 downto 0);
 85:         signal word_addr : std_logic_vector (13 downto 0);
 86:         signal words_read : integer range 0 to 511;
 87:         signal burst_length : integer range 0 to 63;
 88:         signal bram_empty : std_logic;
 89:         signal clear_count : std_logic;
 90:
 91:         signal words_diff : integer range 0 to 511;
 92:
 93:
 94: begin
 95:
 96:         pclk_bufg : BUFG
 97:         port map (
 98:                 O => pclk,
 99:                 I => i_pclk
100:         );
```

```
101:
102:        pclk_n <= not pclk;
103:
104:        vsync_fd : fd
105:        port map (
106:                D => i_vsync,
107:                C => pclk_n,
108:                Q => vsync
109:        );
110:
111:        vsync_d1_fd : fd
112:        port map (
113:                D => vsync,
114:                C => pclk,
115:                Q => vsync_d1
116:        );
117:
118:        frame_count_en <= (not vsync) and vsync_d1;
119:
120:        frame_counter : cpt_upcounter
121:        port map (
122:                i_clk => pclk,
123:                i_enable => frame_count_en,
124:                i_clear => i_reset,
125:                i_preset => '0',
126:                i_lowest => 0,
127:                i_highest => 3,
128:                i_increment => 1,
129:                o_count => o_frame_number,
130:                o_carry => open
131:        );
132:
133:        href_fd : fd
134:        port map (
135:                D => i_href,
136:                C => pclk_n,
137:                Q => href
138:        );
139:
140:        href_d1_fd : fd
141:        port map (
142:                D => href,
143:                C => pclk,
144:                Q => href_d1
145:        );
146:
147:        line_count_en <= href and (not href_d1);
148:
149:        line_counter : cpt_upcounter
150:        generic map (INIT => 0)
151:        port map (
152:                i_clk => pclk,
153:                i_enable => line_count_en,
154:                i_clear => vsync,
155:                i_preset => '0',
156:                i_lowest => 0,
157:                i_highest => 2047,
158:                i_increment => 1,
159:                o_count => o_line_number,
160:                o_carry => open
161:        );
162:
163:
164:        bram_empty <= '1' when (words_diff = 0) else '0';
165:        clear_count <= (bram_empty and (not href)) or i_reset;
166:
167:        --Tracks the number of bytes written BRAM port A
168:        bytes_written_counter : cpt_upcounter
169:        generic map (INIT => 0)
170:        port map (
171:                i_clk => pclk,
172:                i_enable => href,
173:                i_clear => clear_count,
174:                i_preset => '0',
175:                i_lowest => 0,
176:                i_highest => 2047,
177:                i_increment => 1,
178:                o_count => bytes_written,
179:                o_carry => open
180:        );
181:
182:        -- Tracks 32-bit words written to the Block RAM
183:        word_written_calc :
184:                words_written <= (bytes_written / 4);
185:
186:        -- Uses pixel number as the WRITE ADDR for BRAM, shifted up because BRAM port A is set to 8 bits
187:        pixel_addr_calc :
188:                pixel_addr <= std_logic_vector(to_unsigned((bytes_written * 8), pixel_addr'length));
189:
190:        -- Uses word number as the READ ADDR for BRAM, shifted because BRAM port B is set to 32 bits.
191:        word_addr_calc :
192:                word_addr <= std_logic_vector(to_unsigned((words_read * 32), word_addr'length));
193:
194:        process(i_clk)
195:        begin
196:                if ( rising_edge(i_clk) ) then
197:                        data(7 downto 0) <= i_data;
198:                end if;
199:        end process;
200:
```

```
  201:         data(31 downto 8) <= x"000000";
  202:
  203:         pixel_bram : ramb16bwer
  204:                 generic map (
  205:                         DATA_WIDTH_A => 9,
  206:                         DATA_WIDTH_B => 36,
  207:                         SIM_DEVICE => "SPARTAN6",
  208:                         SIM_COLLISION_CHECK => "NONE"
  209:                 )
  210:                 port map (
  211:                         -- Port A Address/Control Signals: 14-bit (each) input: Port A address and control signals
  212:                         ADDRA => pixel_addr,            -- 14-bit input: A port address input
  213:                         WEA => (others => href),                                -- 4-bit input: Port A byte-wide write enable input
  214:                         CLKA => pclk,                           -- 1-bit input: A port clock input
  215:                         ENA => '1',                             -- 1-bit input: A port enable input
  216:                         REGCEA => '0',                  -- 1-bit input: A port register clock enable input
  217:                         RSTA => '0',                            -- 1-bit input: A port register set/reset input
  218:
  219:                         -- Port A Data: 32-bit (each) input: Port A data
  220:                         DIA => data,                    -- 32-bit input: A port data input
  221:                         DIPA => "0000",                                 -- 4-bit input: A port parity input
  222:
  223:                         -- Port A Data: 32-bit (each) output: Port A data
  224:                         DOA => open,                            -- 32-bit output: A port data output
  225:                         DOPA => open,                           -- 4-bit output: A port parity output
  226:
  227:
  228:                         -- Port B Address/Control Signals: 14-bit (each) input: Port B address and control signals
  229:                         ADDRB => word_addr,             -- 14-bit input: B port address input
  230:                         WEB => "0000",                          -- 4-bit input: Port B byte-wide write enable input
  231:                         CLKB => i_clk,                  -- 1-bit input: B port clock input
  232:                         ENB => '1',                             -- 1-bit input: B port enable input
  233:                         REGCEB => '0',                  -- 1-bit input: B port register clock enable input
  234:                         RSTB => '0',                            -- 1-bit input: B port register set/reset input
  235:
  236:                         -- Port B Data: 32-bit (each) input: Port B data
  237:                         DIB => (others => '0'),         -- 32-bit input: B port data input
  238:                         DIPB => "0000", -- 4-bit input: B port parity input
  239:
  240:                         -- Port B Data: 32-bit (each) output: Port B data
  241:                         DOB => o_rd_data,                       -- 32-bit output: B port data output
  242:                         DOPB => open                            -- 4-bit output: B port parity output
  243:         );
  244:
  245:
  246:         o_words_read <= words_read;
  247:
  248:         -- Calculates running total of words (32-bit) stored in BRAM
  249:         words_diff_calc :
  250:                 words_diff <= words_written - words_read; -- written must always be higher than read
  251:
  252:         burst_length_calc :
  253:                 burst_length <= to_integer(unsigned(i_burst_length));
  254:
  255:         -- Compares desired burst to words (32-bit) available to read
  256:         burst_avail_calc :
  257: --              o_burst_available <= '1' when (words_diff > burst_length) or ((words_diff = burst_length) and href = '0') else '0';
  258:                 --o_burst_available <= '1' when (words_diff > burst_length) else '0';-- or ((words_diff = burst_length) and href = '0') else '0';
  259:                 --o_burst_available <= '1' when (words_diff >= burst_length); and words_written /= 0 else '0';-- or ((words_diff = burst_length) and
href = '0') else '0';
  260:                 o_burst_available <= '1' when (words_diff > burst_length) else '0';
  261:
  262:
  263:
  264:         o_collision <= '1' when (pixel_addr(13 downto 5) = word_addr(13 downto 5)) else '0';
  265:
  266:         --Tracks the number of words (32 bit) read from the BRAM
  267:         words_read_counter : cpt_upcounter
  268:         generic map (INIT => 0)
  269:         port map (
  270:                 i_clk => i_clk,
  271:                 i_enable => i_rd_enable,
  272:                 i_clear => clear_count,
  273:                 i_preset => '0',
  274:                 i_lowest => 0,
  275:                 i_highest => 511,
  276:                 i_increment => 1,
  277:                 o_count => words_read,
  278:                 o_carry => open
  279:         );
  280:
  281: end Behavioral;
  282:
```

```
    1: ----------------------------------------------------------------------------------
    2: -- Company:
    3: -- Engineer:
    4: --
    5: -- Create Date:    18:18:05 07/16/2015
    6: -- Design Name:
    7: -- Module Name:    cpt_ovm_mux - Behavioral
    8: -- Project Name:
    9: -- Target Devices:
   10: -- Tool versions:
   11: -- Description:
   12: --
   13: -- Dependencies:
   14: --
   15: -- Revision:
   16: -- Revision 0.01 - File Created
   17: -- Additional Comments:
   18: --
   19: ----------------------------------------------------------------------------------
   20: library IEEE;
   21: use IEEE.STD_LOGIC_1164.ALL;
   22: use IEEE.numeric_std.ALL;
   23:
   24: -- Uncomment the following library declaration if using
   25: -- arithmetic functions with Signed or Unsigned values
   26: --use IEEE.NUMERIC_STD.ALL;
   27:
   28: -- Uncomment the following library declaration if instantiating
   29: -- any Xilinx primitives in this code.
   30: library UNISIM;
   31: use UNISIM.VComponents.all;
   32:
   33:
   34: library mctl;
   35: use mctl.pkg_mctl.all;
   36:
   37: library util;
   38: use util.pkg_util.all;
   39:
   40:
   41: -- Work library for testing
   42: --library work;
   43: --use work.pkg_testing.all;
   44:
   45:
   46: entity cpt_ovm_mux is
   47:         port (
   48:                 i_clk : in std_logic;
   49:                 i_reset : in std_logic;
   50:
   51:                 i0_frame_count : in integer range 0 to 3;
   52:                 i1_frame_count : in integer range 0 to 3;
   53:                 i2_frame_count : in integer range 0 to 3;
   54:                 i3_frame_count : in integer range 0 to 3;
   55:
   56:                 i_frame_addr0 : in std_logic_vector(28 downto 0);
   57:                 i_frame_addr1 : in std_logic_vector(28 downto 0);
   58:                 i_frame_addr2 : in std_logic_vector(28 downto 0);
   59:                 i_frame_addr3 : in std_logic_vector(28 downto 0);
   60:
   61:                 i0_line_offset : in integer range 0 to 2**13-1;
   62:                 i1_line_offset : in integer range 0 to 2**13-1;
   63:                 i2_line_offset : in integer range 0 to 2**13-1;
   64:                 i3_line_offset : in integer range 0 to 2**13-1;
   65:
   66:                 i0_words_read : in integer range 0 to 2**9-1;
   67:                 i1_words_read : in integer range 0 to 2**9-1;
   68:                 i2_words_read : in integer range 0 to 2**9-1;
   69:                 i3_words_read : in integer range 0 to 2**9-1;
   70:
   71:                 i0_line_count : in integer range 0 to 2**9-1;
   72:                 i1_line_count : in integer range 0 to 2**9-1;
   73:                 i2_line_count : in integer range 0 to 2**9-1;
   74:                 i3_line_count : in integer range 0 to 2**9-1;
   75:
   76:                 i0_rd_data : in std_logic_vector(31 downto 0);
   77:                 i1_rd_data : in std_logic_vector(31 downto 0);
   78:                 i2_rd_data : in std_logic_vector(31 downto 0);
   79:                 i3_rd_data : in std_logic_vector(31 downto 0);
   80:
   81:                 i0_burst_available : in std_logic;
   82:                 i1_burst_available : in std_logic;
   83:                 i2_burst_available : in std_logic;
   84:                 i3_burst_available : in std_logic;
   85:
   86:                 o0_rd_enable : out std_logic;
   87:                 o1_rd_enable : out std_logic;
   88:                 o2_rd_enable : out std_logic;
   89:                 o3_rd_enable : out std_logic;
   90:
   91:                 i_burst_length : in std_logic_vector(5 downto 0);
   92:
   93:                 o_mport_miso : out typ_mctl_mport_miso;
   94:                 i_mport_mosi : in typ_mctl_mport_mosi
   95:                 );
   96: end cpt_ovm_mux;
   97:
   98: architecture Behavioral of cpt_ovm_mux is
   99:
  100:         signal burst_length : integer range 0 to 2**6-1;
```

```
101:         signal burst_count : integer range 0 to 2**6-1;
102:         signal burst_enable : std_logic := '0';
103:         signal burst_enable_d1 : std_logic := '0';
104:         signal last_burst_word : std_logic;
105:         signal burst_end : std_logic := '0';
106:         signal burst_done : std_logic;
107:         signal burst_count_enable : std_logic;
108:         signal burst_count_enable_d1 : std_logic;
109:         signal burst_clear : std_logic := '0';
110:         signal burst_available : std_logic;
111:
112:         signal next_camera : std_logic;
113:         signal camera_count_enable : std_logic;
114:         signal camera_count_enable_d1 : std_logic;
115:         signal camera_count_enable_d2 : std_logic;
116:         signal camera_count : integer range 0 to 3;
117:
118:         signal frame_count : integer range 0 to 3;
119:         signal frame_addr : std_logic_vector(28 downto 0);
120:         signal frame_addrnum : integer range 0 to 2**29-1;
121:
122:         signal line_offset : integer range 0 to 2**13-1;
123:
124:         signal words_read : integer range 0 to 2**9-1;
125:         signal line_count : integer range 0 to 2**9-1;
126:         signal word_addr : integer range 0 to 2**27-1;
127:
128:         signal word_addr_frame_addrnum : integer range 0 to 2**29-1;
129:         signal word_addr_line_offset : integer range 0 to 2**29-1;
130:         signal word_addr_words_read : integer range 0 to 2**29-1;
131:         signal word_addr_line_count : integer range 0 to 2**29-1;
132:
133:
134:
135: begin
136:
137:         burst_length <= to_integer(unsigned(i_burst_length));
138:
139:         burst_count_enable <=
140:                 (burst_enable) and
141:                 (not burst_done) and
142:                 (not i_mport_mosi.cmd.full) and
143:                 (not i_mport_mosi.wr.full);
144:
145:         burst_count_enable_d1_fd : fd
146:         port map (
147:                 q => burst_count_enable_d1,
148:                 d => burst_count_enable,
149:                 c => i_clk
150:         );
151:
152:         burst_clear <= not burst_enable;
153:
154:         burst_counter : cpt_upcounter
155:         generic map (INIT => 63)
156:         port map (
157:                 i_clk => i_clk,
158:                 i_enable => burst_count_enable,
159:                 i_clear => burst_clear,
160:                 i_preset => '0',
161:                 i_lowest => 0,
162:                 i_highest => 63,
163:                 i_increment => 1,
164:                 o_count => burst_count,
165:                 o_carry => open
166:         );
167:
168:         last_burst_word <= '1' when burst_count = burst_length else '0';
169:         burst_done <= '1' when burst_count > burst_length else '0';
170:
171:         process(i_clk)
172:         begin
173:                 if ( rising_edge(i_clk) ) then
174:                         if ( last_burst_word = '1' and burst_count_enable = '1' ) then
175:                                 burst_end <= '1';
176:                         else
177:                                 burst_end <= '0';
178:                         end if;
179:                 end if;
180:         end process;
181:
182:         o0_rd_enable <= burst_count_enable when camera_count = 0 else '0';
183:         o1_rd_enable <= burst_count_enable when camera_count = 1 else '0';
184:         o2_rd_enable <= burst_count_enable when camera_count = 2 else '0';
185:         o3_rd_enable <= burst_count_enable when camera_count = 3 else '0';
186:
187:         frame_count_mux:
188:         with camera_count select frame_count <=
189:                 i0_frame_count when 0,
190:                 i1_frame_count when 1,
191:                 i2_frame_count when 2,
192:                 i3_frame_count when 3;
193:
194:
195:         frame_addr_mux:
196:         with frame_count select frame_addr <=
197:                 i_frame_addr0 when 0,
198:                 i_frame_addr1 when 1,
199:                 i_frame_addr2 when 2,
200:                 i_frame_addr3 when 3;
```

```
  201:
  202:         frame_addrnum <= to_integer(unsigned(frame_addr));
  203:
  204:         line_offset_mux:
  205:         with camera_count select line_offset <=
  206:                  i0_line_offset when 0,
  207:                  i1_line_offset when 1,
  208:                  i2_line_offset when 2,
  209:                  i3_line_offset when 3;
  210:
  211:         words_read_mux:
  212:         with camera_count select words_read <=
  213:                 i0_words_read when 0,
  214:                 i1_words_read when 1,
  215:                 i2_words_read when 2,
  216:                 i3_words_read when 3;
  217:
  218:         line_count_mux:
  219:         with camera_count select line_count <=
  220:                 i0_line_count when 0,
  221:                 i1_line_count when 1,
  222:                 i2_line_count when 2,
  223:                 i3_line_count when 3;
  224:
  225:
  226:         word_addr_frame_addrnum <= (frame_addrnum/4);
  227:         word_addr_line_offset <= (512*line_offset);
  228:         word_addr_words_read <= (words_read);
  229:         word_addr_line_count <= (512*line_count);
  230:
  231:         word_addr <= (
  232:                 word_addr_frame_addrnum +
  233:                 word_addr_line_offset +
  234:                 word_addr_words_read +
  235:                 word_addr_line_count
  236:         );
  237:
  238:         process(i_clk)
  239:         begin
  240: --              if ( rising_edge(i_clk) and burst_count = 0 ) then
  241:                 if ( rising_edge(i_clk) and camera_count_enable_d1 = '1' ) then
  242:                         o_mport_miso.cmd.byte_addr <= std_logic_vector(to_unsigned(word_addr, o_mport_miso.cmd.byte_addr'length-2)) & "00";
  243:                 end if;
  244:         end process;
  245:
  246:         o_mport_miso.cmd.en <= burst_end;
  247:         o_mport_miso.cmd.instr <= "010" when burst_end = '1' else "000"; -- write with precharge
  248:         o_mport_miso.cmd.clk <= i_clk;
  249:         o_mport_miso.cmd.bl <= i_burst_length;
  250:
  251:         o_mport_miso.rd.clk <= i_clk;
  252:         o_mport_miso.rd.en <= '0';
  253:
  254:         o_mport_miso.wr.clk <= i_clk;
  255:         o_mport_miso.wr.en <= burst_count_enable_d1;
  256:         --o_mport_miso.wr.en <= burst_count_enable_d1;
  257:         o_mport_miso.wr.mask <= "0000";
  258:
  259:         with camera_count select o_mport_miso.wr.data <=
  260:                 i0_rd_data when 0,
  261:                 i1_rd_data when 1,
  262:                 i2_rd_data when 2,
  263:                 i3_rd_data when 3;
  264:
  265:         burst_available_mux:
  266:         with camera_count select burst_available <=
  267:                 i0_burst_available when 0,
  268:                 i1_burst_available when 1,
  269:                 i2_burst_available when 2,
  270:                 i3_burst_available when 3;
  271:
  272:         next_camera <= burst_done when burst_enable = '1' else '1';
  273:
  274:         --camera_count_enable <= (burst_done and burst_available) or (not burst_available) or (not burst_enable);
  275:         --camera_count_enable <= (burst_done) or ((not camera_count_enable_d1) and (not burst_available));-- or (not burst_enable);
  276:         camera_count_enable <= (not camera_count_enable_d1) and (not camera_count_enable_d2) and (next_camera);--  (burst_done) or ((not camera_count
_enable_d1) and (not burst_available));-- or (not burst_enable);
  277:
  278:
  279:         camera_count_enable_d1_fd : fd
  280:         port map (
  281:                 d => camera_count_enable,
  282:                 q => camera_count_enable_d1,
  283:                 c => i_clk
  284:         );
  285:
  286:         camera_count_enable_d2_fd : fd
  287:         port map (
  288:                 d => camera_count_enable_d1,
  289:                 q => camera_count_enable_d2,
  290:                 c => i_clk
  291:         );
  292:
  293:         process(i_clk, burst_done)
  294:         begin
  295:                 if ( burst_done = '1' or i_reset = '1' ) then
  296:                         burst_enable <= '0';
  297:                 elsif ( rising_edge(i_clk) and camera_count_enable_d1 = '1' ) then
  298:                         burst_enable <= burst_available;
  299:                 end if;
```

```
300:         end process;
301:
302:         burst_enable_d1_fd : fd
303:         port map (
304:                 d => burst_enable,
305:                 q => burst_enable_d1,
306:                 c => i_clk
307:         );
308:
309:         camera_counter : cpt_upcounter
310:         generic map (INIT => 0)
311:         port map (
312:                 i_clk => i_clk,
313:                 i_enable => camera_count_enable,
314:                 i_clear => i_reset,
315:                 i_preset => '0',
316:                 i_lowest => 0,
317:                 i_highest => 3,
318:                 i_increment => 1,
319:                 o_count => camera_count,
320:                 o_carry => open
321:         );
322:
323:
324: end Behavioral;
325:
```

```
  1: --------------------------------------------------------------------------------
  2: -- Company:
  3: -- Engineer:
  4: --
  5: -- Create Date:   15:59:04 07/16/2015
  6: -- Design Name:
  7: -- Module Name:   C:/Xilinx/Projects/bram_buffer/tb_cpt_ovm_bram.vhd
  8: -- Project Name:  bram_buffer
  9: -- Target Device:
 10: -- Tool versions:
 11: -- Description:
 12: --
 13: -- VHDL Test Bench Created by ISE for module: cpt_ovm_bram
 14: --
 15: -- Dependencies:
 16: --
 17: -- Revision:
 18: -- Revision 0.01 - File Created
 19: -- Additional Comments:
 20: --
 21: -- Notes:
 22: -- This testbench has been automatically generated using types std_logic and
 23: -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
 24: -- that these types always be used for the top-level I/O of a design in order
 25: -- to guarantee that the testbench will bind correctly to the post-implementation
 26: -- simulation model.
 27: --------------------------------------------------------------------------------
 28: LIBRARY ieee;
 29: USE ieee.std_logic_1164.ALL;
 30: use ieee.numeric_std.all;
 31:
 32: -- Uncomment the following library declaration if using
 33: -- arithmetic functions with Signed or Unsigned values
 34: --USE ieee.numeric_std.ALL;
 35:
 36: ENTITY tb_cpt_ovm_bram IS
 37: END tb_cpt_ovm_bram;
 38:
 39: ARCHITECTURE behavior OF tb_cpt_ovm_bram IS
 40:
 41:     -- Component Declaration for the Unit Under Test (UUT)
 42:
 43:     COMPONENT cpt_ovm_bram
 44:     PORT(
 45:                         i_pclk : in std_logic;
 46:                         i_vsync : in std_logic;
 47:                         i_href : in std_logic;
 48:                         i_data : in std_logic_vector (7 downto 0);
 49:                         i_reset : in std_logic;
 50:
 51:                         o_rd_data : out std_logic_vector(31 downto 0);
 52:                         o_frame_number : out integer range 0 to 3;
 53:                         o_line_number : out integer range 0 to 2047;
 54:
 55:                         o_words_read: out integer range 0 to 511;
 56:                         i_burst_length : std_logic_vector(5 downto 0);
 57:                         o_burst_available : out std_logic;
 58:                         o_collision : out std_logic;
 59:
 60:                         i_clk : in std_logic;
 61:                         i_rd_enable : in std_logic
 62:         );
 63:     END COMPONENT;
 64:
 65:
 66:    --Inputs
 67:    signal i_pclk : std_logic := '0';
 68:    signal i_vsync : std_logic := '0';
 69:    signal i_href : std_logic := '0';
 70:    signal i_data : std_logic_vector(7 downto 0) := (others => '0');
 71:    signal i_reset : std_logic := '0';
 72:    signal i_burst_length : std_logic_vector(5 downto 0) := (others => '0');
 73:    signal i_clk : std_logic := '0';
 74:    signal i_rd_enable : std_logic := '0';
 75:
 76:         --Outputs
 77:    signal o_rd_data : std_logic_vector(31 downto 0);
 78:    signal o_frame_number : integer range 0 to 3;
 79:    signal o_line_number : integer range 0 to 2047;
 80:    signal o_words_read : integer range 0 to 511;
 81:    signal o_burst_available : std_logic;
 82:
 83:    signal o_collision : std_logic;
 84:
 85:    -- Clock period definitions
 86:    constant i_pclk_period : time := 41.667 ns;
 87:    constant i_clk_period : time := 9.259 ns;
 88:         constant tp : time := 2*i_pclk_period;
 89:         constant tline : time := 780*tp;
 90:
 91: BEGIN
 92:
 93:         -- Instantiate the Unit Under Test (UUT)
 94:    uut: cpt_ovm_bram PORT MAP (
 95:           i_pclk => i_pclk,
 96:           i_vsync => i_vsync,
 97:           i_href => i_href,
 98:           i_data => i_data,
 99:           i_reset => i_reset,
100:           o_rd_data => o_rd_data,
```

```
101:          o_frame_number => o_frame_number,
102:          o_line_number => o_line_number,
103:          o_words_read => o_words_read,
104:          i_burst_length => i_burst_length,
105:          o_burst_available => o_burst_available,
106:          o_collision => o_collision,
107:          i_clk => i_clk,
108:          i_rd_enable => i_rd_enable
109:        );
110:
111:    -- Clock process definitions
112:    i_pclk_process :process
113:    begin
114:                i_pclk <= '0';
115:                wait for i_pclk_period/2;
116:                i_pclk <= '1';
117:                wait for i_pclk_period/2;
118:    end process;
119:
120:    i_clk_process :process
121:    begin
122:                i_clk <= '0';
123:                wait for i_clk_period/2;
124:                i_clk <= '1';
125:                wait for i_clk_period/2;
126:    end process;
127:
128:
129:         vsync_process : process
130:         begin
131:                wait until falling_edge(i_pclk);
132:                i_vsync <= '1';
133:                wait for 4*tline;
134:                i_vsync <= '0';
135:                wait for (512-4)*tline;
136:         end process;
137:
138:         href_process : process
139:         begin
140:                i_href <= '0';
141:                wait for 20*tline;
142:                wait until falling_edge(i_pclk);
143:                for i in 0 to 479 loop
144:                       i_href <= '1';
145:                       wait for 640*tp;
146:                       i_href <= '0';
147:                       wait for 140*tp;
148:                end loop;
149:                wait for 12*tline;
150:         end process;
151:
152:         data_process : process
153:         begin
154:                wait until rising_edge(i_href);
155:                --              wait until falling_edge(i_pclk);
156:                for i in 0 to 1279 loop
157:                       i_data <= std_logic_vector(to_unsigned(i mod 256, i_data'length)); --mod to avoid truncation warnings everywhere
158:                       wait for i_pclk_period;
159:                end loop;
160:         end process;
161:
162:
163:         --i_rd_enable <= o_burst_available;
164:
165:
166:         rd_proc: process
167:    begin
168:
169:                wait until rising_edge(i_clk);
170:                if ( o_burst_available = '1' ) then
171:
172:                       for i in 0 to 15 loop
173:
174:                              wait until rising_edge(i_clk);
175:                              i_rd_enable <= '1';
176:
177:                       end loop;
178:
179:
180:                       wait until rising_edge(i_clk);
181:                       i_rd_enable <= '0';
182:
183:                end if;
184:
185:    end process;
186:
187:
188:
189:         i_burst_length <= "010000";
190:
191:    -- Stimulus process
192:    stim_proc: process
193:    begin
194:       -- hold reset state for 100 ns.
195:                i_reset <= '1';
196:
197:       wait for 100 ns;
198:                i_reset <= '0';
199:
200:
```

```
  201:       wait for i_pclk_period*10;
  202:
  203:       -- insert stimulus here
  204:
  205:       wait;
  206:    end process;
  207:
  208: END;
```

```
  1:
  2:
  3: LIBRARY ieee;
  4: USE ieee.std_logic_1164.ALL;
  5: USE ieee.numeric_std.ALL;
  6:
  7:
  8: -- Work library for testing
  9: library work;
 10: use work.pkg_testing.all;
 11:
 12:
 13:   ENTITY test_bram_to_mux IS
 14:   END test_bram_to_mux;
 15:
 16:   ARCHITECTURE behavior OF test_bram_to_mux IS
 17:
 18:
 19:         component sim_ovm_testing is
 20:                 generic (
 21:                         SMALL_FRAME : string := "FALSE"
 22:                 );
 23:                 port (
 24:                         i_ovm_sccb_mosi : in typ_ovm_sccb_mosi;
 25:                         io_ovm_sccb_bidir : inout typ_ovm_sccb_bidir;
 26:                         o_ovm_video_miso : out typ_ovm_video_miso
 27:                 );
 28:         end component;
 29:
 30:
 31:    COMPONENT cpt_ovm_bram
 32:     PORT(
 33:                         i_pclk : in std_logic;
 34:                         i_vsync : in std_logic;
 35:                         i_href : in std_logic;
 36:                         i_data : in std_logic_vector (7 downto 0);
 37:                         i_reset : in std_logic;
 38:
 39:                         o_rd_data : out std_logic_vector(31 downto 0);
 40:                         o_frame_number : out integer range 0 to 3;
 41:                         o_line_number : out integer range 0 to 2047;
 42:
 43:                         o_words_read: out integer range 0 to 511;
 44:                         i_burst_length : std_logic_vector(5 downto 0);
 45:                         o_burst_available : out std_logic;
 46:                         o_collision : out std_logic;
 47:
 48:                         i_clk : in std_logic;
 49:                         i_rd_enable : in std_logic
 50:         );
 51:     END COMPONENT;
 52:
 53:
 54:     COMPONENT cpt_ovm_mux
 55:     PORT(
 56:          i_clk : IN  std_logic;
 57:          i_reset : IN  std_logic;
 58:          i0_frame_count : IN  integer range 0 to 3;
 59:          i1_frame_count : IN  integer range 0 to 3;
 60:          i2_frame_count : IN  integer range 0 to 3;
 61:          i3_frame_count : IN  integer range 0 to 3;
 62:          i_frame_addr0 : IN  std_logic_vector(25 downto 0);
 63:          i_frame_addr1 : IN  std_logic_vector(25 downto 0);
 64:          i_frame_addr2 : IN  std_logic_vector(25 downto 0);
 65:          i_frame_addr3 : IN  std_logic_vector(25 downto 0);
 66:          i0_line_offset : IN  integer range 0 to 8191;
 67:          i1_line_offset : IN  integer range 0 to 8191;
 68:          i2_line_offset : IN  integer range 0 to 8191;
 69:          i3_line_offset : IN  integer range 0 to 8191;
 70:          i0_words_read : IN  integer range 0 to 511;
 71:          i1_words_read : IN  integer range 0 to 511;
 72:          i2_words_read : IN  integer range 0 to 511;
 73:          i3_words_read : IN  integer range 0 to 511;
 74:          i0_line_count : IN  integer range 0 to 511;
 75:          i1_line_count : IN  integer range 0 to 511;
 76:          i2_line_count : IN  integer range 0 to 511;
 77:          i3_line_count : IN  integer range 0 to 511;
 78:          i0_rd_data : IN  std_logic_vector(31 downto 0);
 79:          i1_rd_data : IN  std_logic_vector(31 downto 0);
 80:          i2_rd_data : IN  std_logic_vector(31 downto 0);
 81:          i3_rd_data : IN  std_logic_vector(31 downto 0);
 82:          i0_burst_available : IN  std_logic;
 83:          i1_burst_available : IN  std_logic;
 84:          i2_burst_available : IN  std_logic;
 85:          i3_burst_available : IN  std_logic;
 86:          o0_rd_enable : OUT  std_logic;
 87:          o1_rd_enable : OUT  std_logic;
 88:          o2_rd_enable : OUT  std_logic;
 89:          o3_rd_enable : OUT  std_logic;
 90:          i_burst_length : IN  std_logic_vector(5 downto 0);
 91:          o_mport_miso : OUT  typ_mctl_mport_miso;
 92:          i_mport_mosi : IN  typ_mctl_mport_mosi
 93:         );
 94:     END COMPONENT;
 95:
 96:
 97:    signal mport_mosi : typ_mctl_mport_mosi := init_mctl_mport_mosi;
 98:    signal mport_miso : typ_mctl_mport_miso := init_mctl_mport_miso;
 99:
100:    signal clk : std_logic := '0';
```

```
101:    signal bram_reset : std_logic := '0';
102:    signal mux_reset : std_logic := '0';
103:    signal burst_length : std_logic_vector(5 downto 0) := std_logic_vector(to_unsigned(15,6));
104:
105:         signal frame_addr0 : std_logic_vector(25 downto 0) := "00" & x"000000";
106:    signal frame_addr1 : std_logic_vector(25 downto 0) := "00" & x"100000";
107:    signal frame_addr2 : std_logic_vector(25 downto 0) := "00" & x"200000";
108:    signal frame_addr3 : std_logic_vector(25 downto 0) := "00" & x"300000";
109:
110:    signal line_offset0 : integer range 0 to 8191 := 0;
111:    signal line_offset1 : integer range 0 to 8191 := 1024;
112:    signal line_offset2 : integer range 0 to 8191 := 1024;
113:    signal line_offset3 : integer range 0 to 8191 := 0;
114:
115:         signal ovm0_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
116:         signal ovm0_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
117:         signal ovm0_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
118:         --signal ovm0.href : std_logic := '0';
119:    signal ovm0_bram_rd_enable : std_logic := '0';
120:    signal ovm0_bram_rd_data : std_logic_vector(31 downto 0);
121:    signal ovm0_bram_frame_number : integer range 0 to 3;
122:    signal ovm0_bram_line_number : integer range 0 to 2047;
123:    signal ovm0_bram_words_read : integer range 0 to 511;
124:    signal ovm0_bram_burst_available : std_logic;
125:
126:         signal ovm1_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
127:         signal ovm1_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
128:         signal ovm1_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
129:    signal ovm1_bram_rd_enable : std_logic := '0';
130:    signal ovm1_bram_rd_data : std_logic_vector(31 downto 0);
131:    signal ovm1_bram_frame_number : integer range 0 to 3;
132:    signal ovm1_bram_line_number : integer range 0 to 2047;
133:    signal ovm1_bram_words_read : integer range 0 to 511;
134:    signal ovm1_bram_burst_available : std_logic;
135:
136:         signal ovm2_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
137:         signal ovm2_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
138:         signal ovm2_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
139:    signal ovm2_bram_rd_enable : std_logic := '0';
140:    signal ovm2_bram_rd_data : std_logic_vector(31 downto 0);
141:    signal ovm2_bram_frame_number : integer range 0 to 3;
142:    signal ovm2_bram_line_number : integer range 0 to 2047;
143:    signal ovm2_bram_words_read : integer range 0 to 511;
144:    signal ovm2_bram_burst_available : std_logic;
145:
146:         signal ovm3_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
147:         signal ovm3_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
148:         signal ovm3_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
149:    signal ovm3_bram_rd_enable : std_logic := '0';
150:    signal ovm3_bram_rd_data : std_logic_vector(31 downto 0);
151:    signal ovm3_bram_frame_number : integer range 0 to 3;
152:    signal ovm3_bram_line_number : integer range 0 to 2047;
153:    signal ovm3_bram_words_read : integer range 0 to 511;
154:    signal ovm3_bram_burst_available : std_logic;
155:
156:    signal o0_collision : std_logic;
157:    signal o1_collision : std_logic;
158:    signal o2_collision : std_logic;
159:    signal o3_collision : std_logic;
160:
161:
162:    -- Clock period definitions
163:    constant xvclk_period : time := 166.667 ns;
164:   -- constant pclk_period : time := 41.667 ns;
165:    constant clk_period : time := 9.259 ns;
166:         --constant tp : time := 2*pclk_period;
167:         --constant tline : time := 780*tp;
168:
169:         constant SMALL_FRAME : string := "FALSE";
170:
171: BEGIN
172:
173:
174:         ovm0_sccb_mosi.pwdn <= '0';
175:         ovm1_sccb_mosi.pwdn <= '0';
176:         ovm2_sccb_mosi.pwdn <= '0';
177:         ovm3_sccb_mosi.pwdn <= '0';
178:
179:    ovm0_xvclk_process : process
180:    begin
181:                 ovm0_sccb_mosi.xvclk <= '0';
182:                 wait for xvclk_period/2;
183:                 ovm0_sccb_mosi.xvclk <= '1';
184:                 wait for xvclk_period/2;
185:    end process;
186:
187:    ovm1_xvclk_process : process
188:    begin
189:                 ovm1_sccb_mosi.xvclk <= '0';
190:                 wait for xvclk_period/2;
191:                 ovm1_sccb_mosi.xvclk <= '1';
192:                 wait for xvclk_period/2;
193:    end process;
194:
195:    ovm2_xvclk_process : process
196:    begin
197:                 ovm2_sccb_mosi.xvclk <= '0';
198:                 wait for xvclk_period/2;
199:                 ovm2_sccb_mosi.xvclk <= '1';
200:                 wait for xvclk_period/2;
```

```
201:    end process;
202:
203:    ovm3_xvclk_process : process
204:    begin
205:                ovm3_sccb_mosi.xvclk <= '0';
206:                wait for xvclk_period/2;
207:                ovm3_sccb_mosi.xvclk <= '1';
208:                wait for xvclk_period/2;
209:    end process;
210:
211:
212:         ovm0 : sim_ovm_testing
213:         generic map (
214:                 SMALL_FRAME => SMALL_FRAME
215:         )
216:         port map (
217:                 i_ovm_sccb_mosi => ovm0_sccb_mosi,
218:                 io_ovm_sccb_bidir => ovm0_sccb_bidir,
219:                 o_ovm_video_miso => ovm0_video_miso
220:         );
221:
222:         ovm1 : sim_ovm_testing
223:         generic map (
224:                 SMALL_FRAME => SMALL_FRAME
225:         )
226:         port map (
227:                 i_ovm_sccb_mosi => ovm1_sccb_mosi,
228:                 io_ovm_sccb_bidir => ovm1_sccb_bidir,
229:                 o_ovm_video_miso => ovm1_video_miso
230:         );
231:
232:         ovm2 : sim_ovm_testing
233:         generic map (
234:                 SMALL_FRAME => SMALL_FRAME
235:         )
236:         port map (
237:                 i_ovm_sccb_mosi => ovm2_sccb_mosi,
238:                 io_ovm_sccb_bidir => ovm2_sccb_bidir,
239:                 o_ovm_video_miso => ovm2_video_miso
240:         );
241:
242:         ovm3 : sim_ovm_testing
243:         generic map (
244:                 SMALL_FRAME => SMALL_FRAME
245:         )
246:         port map (
247:                 i_ovm_sccb_mosi => ovm3_sccb_mosi,
248:                 io_ovm_sccb_bidir => ovm3_sccb_bidir,
249:                 o_ovm_video_miso => ovm3_video_miso
250:         );
251:
252:
253:         -- Instantiate the Unit Under Test (UUT)
254:    ovm_mux: cpt_ovm_mux PORT MAP (
255:                  i_clk => clk,
256:                  i_reset => mux_reset,
257:                  i0_frame_count => ovm0_bram_frame_number,
258:                  i1_frame_count => ovm1_bram_frame_number,
259:                  i2_frame_count => ovm2_bram_frame_number,
260:                  i3_frame_count => ovm3_bram_frame_number,
261:                  i_frame_addr0 => frame_addr0,
262:                  i_frame_addr1 => frame_addr1,
263:                  i_frame_addr2 => frame_addr2,
264:                  i_frame_addr3 => frame_addr3,
265:                  i0_line_offset => line_offset0,
266:                  i1_line_offset => line_offset1,
267:                  i2_line_offset => line_offset2,
268:                  i3_line_offset => line_offset3,
269:                  i0_words_read => ovm0_bram_words_read,
270:                  i1_words_read => ovm1_bram_words_read,
271:                  i2_words_read => ovm2_bram_words_read,
272:                  i3_words_read => ovm3_bram_words_read,
273:                  i0_line_count => ovm0_bram_line_number,
274:                  i1_line_count => ovm1_bram_line_number,
275:                  i2_line_count => ovm2_bram_line_number,
276:                  i3_line_count => ovm3_bram_line_number,
277:                  i0_rd_data => ovm0_bram_rd_data,
278:                  i1_rd_data => ovm1_bram_rd_data,
279:                  i2_rd_data => ovm2_bram_rd_data,
280:                  i3_rd_data => ovm3_bram_rd_data,
281:                  i0_burst_available => ovm0_bram_burst_available,
282:                  i1_burst_available => ovm1_bram_burst_available,
283:                  i2_burst_available => ovm2_bram_burst_available,
284:                  i3_burst_available => ovm3_bram_burst_available,
285:                  o0_rd_enable => ovm0_bram_rd_enable,
286:                  o1_rd_enable => ovm1_bram_rd_enable,
287:                  o2_rd_enable => ovm2_bram_rd_enable,
288:                  o3_rd_enable => ovm3_bram_rd_enable,
289:                  i_burst_length => burst_length,
290:                  o_mport_miso => mport_miso,
291:                  i_mport_mosi => mport_mosi
292:           );
293:
294:
295:         ovm0_bram : cpt_ovm_bram PORT MAP (
296:                 i_pclk => ovm0_video_miso.pclk,
297:                 i_vsync => ovm0_video_miso.vsync,
298:                 i_href => ovm0_video_miso.href,
299:                 i_data => ovm0_video_miso.data,
300:                 i_reset => bram_reset,
```

```
301:                 o_rd_data => ovm0_bram_rd_data,
302:                 o_frame_number => ovm0_bram_frame_number,
303:                 o_line_number => ovm0_bram_line_number,
304:                 o_words_read => ovm0_bram_words_read,
305:                 i_burst_length => burst_length,
306:                 o_burst_available => ovm0_bram_burst_available,
307:                 o_collision => o0_collision,
308:                 i_clk => clk,
309:                 i_rd_enable => ovm0_bram_rd_enable
310:         );
311:
312:         ovm1_bram : cpt_ovm_bram PORT MAP (
313:                 i_pclk => ovm1_video_miso.pclk,
314:                 i_vsync => ovm1_video_miso.vsync,
315:                 i_href => ovm1_video_miso.href,
316:                 i_data => ovm1_video_miso.data,
317:                 i_reset => bram_reset,
318:                 o_rd_data => ovm1_bram_rd_data,
319:                 o_frame_number => ovm1_bram_frame_number,
320:                 o_line_number => ovm1_bram_line_number,
321:                 o_words_read => ovm1_bram_words_read,
322:                 i_burst_length => burst_length,
323:                 o_burst_available => ovm1_bram_burst_available,
324:                 o_collision => o1_collision,
325:                 i_clk => clk,
326:                 i_rd_enable => ovm1_bram_rd_enable
327:         );
328:
329:         ovm2_bram : cpt_ovm_bram PORT MAP (
330:                 i_pclk => ovm2_video_miso.pclk,
331:                 i_vsync => ovm2_video_miso.vsync,
332:                 i_href => ovm2_video_miso.href,
333:                 i_data => ovm2_video_miso.data,
334:                 i_reset => bram_reset,
335:                 o_rd_data => ovm2_bram_rd_data,
336:                 o_frame_number => ovm2_bram_frame_number,
337:                 o_line_number => ovm2_bram_line_number,
338:                 o_words_read => ovm2_bram_words_read,
339:                 i_burst_length => burst_length,
340:                 o_burst_available => ovm2_bram_burst_available,
341:                 o_collision => o2_collision,
342:                 i_clk => clk,
343:                 i_rd_enable => ovm2_bram_rd_enable
344:         );
345:
346:         ovm3_bram: cpt_ovm_bram PORT MAP (
347:                 i_pclk => ovm3_video_miso.pclk,
348:                 i_vsync => ovm3_video_miso.vsync,
349:                 i_href => ovm3_video_miso.href,
350:                 i_data => ovm3_video_miso.data,
351:                 i_reset => bram_reset,
352:                 o_rd_data => ovm3_bram_rd_data,
353:                 o_frame_number => ovm3_bram_frame_number,
354:                 o_line_number => ovm3_bram_line_number,
355:                 o_words_read => ovm3_bram_words_read,
356:                 i_burst_length => burst_length,
357:                 o_burst_available => ovm3_bram_burst_available,
358:                 o_collision => o3_collision,
359:                 i_clk => clk,
360:                 i_rd_enable => ovm3_bram_rd_enable
361:         );
362:
363:
364:     -- Clock process definitions
365:     i_clk_process :process
366:     begin
367:                 clk <= '0';
368:                 wait for clk_period/2;
369:                 clk <= '1';
370:                 wait for clk_period/2;
371:     end process;
372:
373:
374: --    -- Clock process definitions
375: --    rd_full_process :process
376: --    begin
377: --                wait until rising_edge(i_clk);
378: --                mport_mosi.rd.full <= '0';
379: --                wait for clk_period * 10;
380: --                wait until rising_edge(i_clk);
381: --                mport_mosi.rd.full <= '1';
382: --                wait for clk_period * 10;
383: --    end process;
384: --
385: --
386: --       -- Clock process definitions
387: --    cmd_full_process :process
388: --    begin
389: --                wait until rising_edge(i_clk);
390: --                mport_mosi.cmd.full <= '0';
391: --                wait for clk_period * 100;
392: --                wait until rising_edge(i_clk);
393: --                mport_mosi.cmd.full <= '1';
394: --                wait for clk_period * 100;
395: --    end process;
396:
397:
398:
399: --       -- Clock process definitions
400: --    ovm0_pclk_process :process
```

```
401: --   begin
402: --             ovm0.pclk <= '0';
403: --             wait for pclk_period/2;
404: --             ovm0.pclk <= '1';
405: --             wait for pclk_period/2;
406: --   end process;
407: --
408: --   ovm1_pclk_process :process
409: --   begin
410: --             ovm1.pclk <= '0';
411: --             wait for pclk_period/2;
412: --             ovm1.pclk <= '1';
413: --             wait for pclk_period/2;
414: --   end process;
415: --
416: --   ovm2_pclk_process :process
417: --   begin
418: --             ovm2.pclk <= '0';
419: --             wait for pclk_period/2;
420: --             ovm2.pclk <= '1';
421: --             wait for pclk_period/2;
422: --   end process;
423: --
424: --   ovm3_pclk_process :process
425: --   begin
426: --             ovm3.pclk <= '0';
427: --             wait for pclk_period/2;
428: --             ovm3.pclk <= '1';
429: --             wait for pclk_period/2;
430: --   end process;
431: --
432: --
433: --      ovm0_vsync_process : process
434: --      begin
435: --             wait until falling_edge(ovm0.pclk);
436: --             ovm0.vsync <= '1';
437: --             wait for 4*tline;
438: --             ovm0.vsync <= '0';
439: --             wait for (512-4)*tline;
440: --      end process;
441: --
442: --      ovm1_vsync_process : process
443: --      begin
444: --             wait until falling_edge(ovm1.pclk);
445: --             ovm1.vsync <= '1';
446: --             wait for 4*tline;
447: --             ovm1.vsync <= '0';
448: --             wait for (512-4)*tline;
449: --      end process;
450: --
451: --      ovm2_vsync_process : process
452: --      begin
453: --             wait until falling_edge(ovm2.pclk);
454: --             ovm2.vsync <= '1';
455: --             wait for 4*tline;
456: --             ovm2.vsync <= '0';
457: --             wait for (512-4)*tline;
458: --      end process;
459: --
460: --      ovm3_vsync_process : process
461: --      begin
462: --             wait until falling_edge(ovm3.pclk);
463: --             ovm3.vsync <= '1';
464: --             wait for 4*tline;
465: --             ovm3.vsync <= '0';
466: --             wait for (512-4)*tline;
467: --      end process;
468: --
469: --      ovm0_href_process : process
470: --      begin
471: --             ovm0.href <= '0';
472: --             wait for 20*tline;
473: --             wait until falling_edge(ovm0.pclk);
474: --             for i in 0 to 479 loop
475: --                    ovm0.href <= '1';
476: --                    wait for 640*tp;
477: --                    ovm0.href <= '0';
478: --                    wait for 140*tp;
479: --             end loop;
480: --             wait for 12*tline;
481: --      end process;
482: --
483: --      ovm1_href_process : process
484: --      begin
485: --             ovm1.href <= '0';
486: --             wait for 20*tline;
487: --             wait until falling_edge(ovm1.pclk);
488: --             for i in 0 to 479 loop
489: --                    ovm1.href <= '1';
490: --                    wait for 640*tp;
491: --                    ovm1.href <= '0';
492: --                    wait for 140*tp;
493: --             end loop;
494: --             wait for 12*tline;
495: --      end process;
496: --
497: --      ovm2_href_process : process
498: --      begin
499: --             ovm2.href <= '0';
500: --             wait for 20*tline;
```

```
501: --              wait until falling_edge(ovm2.pclk);
502: --              for i in 0 to 479 loop
503: --                      ovm2.href <= '1';
504: --                      wait for 640*tp;
505: --                      ovm2.href <= '0';
506: --                      wait for 140*tp;
507: --              end loop;
508: --              wait for 12*tline;
509: --      end process;
510: --
511: --      ovm3_href_process : process
512: --      begin
513: --              ovm3.href <= '0';
514: --              wait for 20*tline;
515: --              wait until falling_edge(ovm3.pclk);
516: --              for i in 0 to 479 loop
517: --                      ovm3.href <= '1';
518: --                      wait for 640*tp;
519: --                      ovm3.href <= '0';
520: --                      wait for 140*tp;
521: --              end loop;
522: --              wait for 12*tline;
523: --      end process;
524: --
525: --      ovm0_data_process : process
526: --      begin
527: --              --wait until rising_edge(ovm0.href);
528: --              wait until ovm0.href = '1';
529: --              report "OVM0 rise";
530: --              --              wait until falling_edge(i_pclk);
531: --              for i in 0 to 1279 loop
532: --                      ovm0.data <= std_logic_vector(to_unsigned(i mod 256, ovm0.data'length)); --mod to avoid truncation warnings everywhere
533: --                      wait for pclk_period;
534: --              end loop;
535: --      end process;
536: --
537: --      ovm1_data_process : process
538: --      begin
539: --              --wait until rising_edge(ovm1.href);
540: --              wait until ovm1.href = '1';
541: --              --              wait until falling_edge(i_pclk);
542: --              for i in 0 to 1279 loop
543: --                      ovm1.data <= std_logic_vector(to_unsigned(i mod 256, ovm1.data'length)); --mod to avoid truncation warnings everywhere
544: --                      wait for pclk_period;
545: --              end loop;
546: --      end process;
547: --
548: --      ovm2_data_process : process
549: --      begin
550: --              wait until rising_edge(ovm2.href);
551: --              --              wait until falling_edge(i_pclk);
552: --              for i in 0 to 1279 loop
553: --                      ovm2.data <= std_logic_vector(to_unsigned(i mod 256, ovm2.data'length)); --mod to avoid truncation warnings everywhere
554: --                      wait for pclk_period;
555: --              end loop;
556: --      end process;
557: --
558: --      ovm3_data_process : process
559: --      begin
560: --              wait until rising_edge(ovm3.href);
561: --              --              wait until falling_edge(i_pclk);
562: --              for i in 0 to 1279 loop
563: --                      ovm3.data <= std_logic_vector(to_unsigned(i mod 256, ovm3.data'length)); --mod to avoid truncation warnings everywhere
564: --                      wait for pclk_period;
565: --              end loop;
566: --      end process;
567:
568:
569:
570:     -- Clock process definitions
571:     rd_full_process :process
572:     begin
573:                 wait until rising_edge(clk);
574:                 mport_mosi.wr.full <= '0';
575:                 wait for clk_period * 10;
576:                 wait until rising_edge(clk);
577:                 mport_mosi.wr.full <= '1';
578:                 wait for clk_period * 10;
579:     end process;
580:
581:
582:          -- Clock process definitions
583:     cmd_full_process :process
584:     begin
585:                 wait until rising_edge(clk);
586:                 mport_mosi.cmd.full <= '0';
587:                 wait for clk_period * 100;
588:                 wait until rising_edge(clk);
589:                 mport_mosi.cmd.full <= '1';
590:                 wait for clk_period * 100;
591:     end process;
592:
593:
594:
595:
596:     stim_proc: process
597:     begin
598:                 bram_reset <= '0';
599:                 mux_reset <= '0';
600:        wait for 100 ns;
```

```
601:                bram_reset <= '1';
602:                mux_reset <= '1';
603:       wait for 100 ns;
604:                bram_reset <= '0';
605:                mux_reset <= '0';
606:       wait;
607:    end process;
608:
609: END;
```

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   00:42:07 07/17/2015
-- Design Name:
-- Module Name:   C:/Users/Chris/Dropbox/Capstone/QuadCam/QuadCamVHDL/src/cam/test_ovm_mux.vhd
-- Project Name:  QuadCamVHDL
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: cpt_ovm_mux
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;


-- Work library for testing
library work;
use work.pkg_testing.all;

ENTITY test_ovm_mux IS
END test_ovm_mux;

ARCHITECTURE behavior OF test_ovm_mux IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT cpt_ovm_mux
    PORT(
         i_clk : IN  std_logic;
         i_reset : IN  std_logic;
         i0_frame_count : IN  integer range 0 to 3;
         i1_frame_count : IN  integer range 0 to 3;
         i2_frame_count : IN  integer range 0 to 3;
         i3_frame_count : IN  integer range 0 to 3;
         i_frame_addr0 : IN  std_logic_vector(25 downto 0);
         i_frame_addr1 : IN  std_logic_vector(25 downto 0);
         i_frame_addr2 : IN  std_logic_vector(25 downto 0);
         i_frame_addr3 : IN  std_logic_vector(25 downto 0);
         i0_line_offset : IN  integer range 0 to 8191;
         i1_line_offset : IN  integer range 0 to 8191;
         i2_line_offset : IN  integer range 0 to 8191;
         i3_line_offset : IN  integer range 0 to 8191;
         i0_words_read : IN  integer range 0 to 511;
         i1_words_read : IN  integer range 0 to 511;
         i2_words_read : IN  integer range 0 to 511;
         i3_words_read : IN  integer range 0 to 511;
         i0_line_count : IN  integer range 0 to 511;
         i1_line_count : IN  integer range 0 to 511;
         i2_line_count : IN  integer range 0 to 511;
         i3_line_count : IN  integer range 0 to 511;
         i0_rd_data : IN  std_logic_vector(31 downto 0);
         i1_rd_data : IN  std_logic_vector(31 downto 0);
         i2_rd_data : IN  std_logic_vector(31 downto 0);
         i3_rd_data : IN  std_logic_vector(31 downto 0);
         i0_burst_available : IN  std_logic;
         i1_burst_available : IN  std_logic;
         i2_burst_available : IN  std_logic;
         i3_burst_available : IN  std_logic;
         o0_rd_enable : OUT  std_logic;
         o1_rd_enable : OUT  std_logic;
         o2_rd_enable : OUT  std_logic;
         o3_rd_enable : OUT  std_logic;
         i_burst_length : IN  std_logic_vector(5 downto 0);
         o_mport_miso : OUT  typ_mctl_mport_miso;
         i_mport_mosi : IN  typ_mctl_mport_mosi
        );
    END COMPONENT;


   --Inputs
   signal i_clk : std_logic := '0';
   signal i_reset : std_logic := '0';
   signal i0_frame_count : integer range 0 to 3 := 0;
   signal i1_frame_count : integer range 0 to 3 := 0;
   signal i2_frame_count : integer range 0 to 3 := 0;
   signal i3_frame_count : integer range 0 to 3 := 0;
   signal i_frame_addr0 : std_logic_vector(25 downto 0) := "00" & x"000000";
   signal i_frame_addr1 : std_logic_vector(25 downto 0) := "00" & x"100000";
   signal i_frame_addr2 : std_logic_vector(25 downto 0) := "00" & x"200000";
```

```
  101:    signal i_frame_addr3 : std_logic_vector(25 downto 0) := "00" & x"300000";
  102:    signal i0_line_offset : integer range 0 to 8191 := 0;
  103:    signal i1_line_offset : integer range 0 to 8191 := 1024;
  104:    signal i2_line_offset : integer range 0 to 8191 := 1024;
  105:    signal i3_line_offset : integer range 0 to 8191 := 0;
  106:    signal i0_words_read : integer range 0 to 511 := 0;
  107:    signal i1_words_read : integer range 0 to 511 := 0;
  108:    signal i2_words_read : integer range 0 to 511 := 0;
  109:    signal i3_words_read : integer range 0 to 511 := 0;
  110:    signal i0_line_count : integer range 0 to 511 := 0;
  111:    signal i1_line_count : integer range 0 to 511 := 0;
  112:    signal i2_line_count : integer range 0 to 511 := 0;
  113:    signal i3_line_count : integer range 0 to 511 := 0;
  114:    signal i0_rd_data : std_logic_vector(31 downto 0) := x"00000000";
  115:    signal i1_rd_data : std_logic_vector(31 downto 0) := x"11111111";
  116:    signal i2_rd_data : std_logic_vector(31 downto 0) := x"22222222";
  117:    signal i3_rd_data : std_logic_vector(31 downto 0) := x"33333333";
  118:    signal i0_burst_available : std_logic := '1';
  119:    signal i1_burst_available : std_logic := '1';
  120:    signal i2_burst_available : std_logic := '1';
  121:    signal i3_burst_available : std_logic := '1';
  122:    signal i_burst_length : std_logic_vector(5 downto 0) := std_logic_vector(to_unsigned(15, 6)); -- Actual burst length = 16, but RAM adds 1 automati
cally
  123:
  124:    signal i_mport_mosi : typ_mctl_mport_mosi := init_mctl_mport_mosi;
  125:
  126:        --Outputs
  127:    signal o0_rd_enable : std_logic;
  128:    signal o1_rd_enable : std_logic;
  129:    signal o2_rd_enable : std_logic;
  130:    signal o3_rd_enable : std_logic;
  131:
  132:    signal o_mport_miso : typ_mctl_mport_miso := init_mctl_mport_miso;
  133:
  134:        -- Clock period definitions
  135:    constant i_clk_period : time := 9.259 ns;
  136:
  137: BEGIN
  138:
  139:        -- Instantiate the Unit Under Test (UUT)
  140:    uut: cpt_ovm_mux PORT MAP (
  141:          i_clk => i_clk,
  142:          i_reset => i_reset,
  143:          i0_frame_count => i0_frame_count,
  144:          i1_frame_count => i1_frame_count,
  145:          i2_frame_count => i2_frame_count,
  146:          i3_frame_count => i3_frame_count,
  147:          i_frame_addr0 => i_frame_addr0,
  148:          i_frame_addr1 => i_frame_addr1,
  149:          i_frame_addr2 => i_frame_addr2,
  150:          i_frame_addr3 => i_frame_addr3,
  151:          i0_line_offset => i0_line_offset,
  152:          i1_line_offset => i1_line_offset,
  153:          i2_line_offset => i2_line_offset,
  154:          i3_line_offset => i3_line_offset,
  155:          i0_words_read => i0_words_read,
  156:          i1_words_read => i1_words_read,
  157:          i2_words_read => i2_words_read,
  158:          i3_words_read => i3_words_read,
  159:          i0_line_count => i0_line_count,
  160:          i1_line_count => i1_line_count,
  161:          i2_line_count => i2_line_count,
  162:          i3_line_count => i3_line_count,
  163:          i0_rd_data => i0_rd_data,
  164:          i1_rd_data => i1_rd_data,
  165:          i2_rd_data => i2_rd_data,
  166:          i3_rd_data => i3_rd_data,
  167:          i0_burst_available => i0_burst_available,
  168:          i1_burst_available => i1_burst_available,
  169:          i2_burst_available => i2_burst_available,
  170:          i3_burst_available => i3_burst_available,
  171:          o0_rd_enable => o0_rd_enable,
  172:          o1_rd_enable => o1_rd_enable,
  173:          o2_rd_enable => o2_rd_enable,
  174:          o3_rd_enable => o3_rd_enable,
  175:          i_burst_length => i_burst_length,
  176:          o_mport_miso => o_mport_miso,
  177:          i_mport_mosi => i_mport_mosi
  178:        );
  179:
  180:    -- Clock process definitions
  181:    i_clk_process :process
  182:    begin
  183:                i_clk <= '0';
  184:                wait for i_clk_period/2;
  185:                i_clk <= '1';
  186:                wait for i_clk_period/2;
  187:    end process;
  188:
  189:
  190:    -- Clock process definitions
  191:    rd_full_process :process
  192:    begin
  193:                wait until rising_edge(i_clk);
  194:                i_mport_mosi.rd.full <= '0';
  195:                wait for i_clk_period * 10;
  196:                wait until rising_edge(i_clk);
  197:                i_mport_mosi.rd.full <= '1';
  198:                wait for i_clk_period * 10;
  199:    end process;
```

```
200:
201:
202:         -- Clock process definitions
203:    cmd_full_process :process
204:    begin
205:                wait until rising_edge(i_clk);
206:                i_mport_mosi.cmd.full <= '0';
207:                wait for i_clk_period * 100;
208:                wait until rising_edge(i_clk);
209:                i_mport_mosi.cmd.full <= '1';
210:                wait for i_clk_period * 100;
211:    end process;
212:
213:
214:
215:    -- Stimulus process
216:    stim_proc: process
217:    begin
218:       -- hold reset state for 100 ns.
219:                i_reset <= '1';
220:       wait for 100 ns;
221:                i_reset <= '0';
222:
223:       wait for i_clk_period*10;
224:
225:       -- insert stimulus here
226:
227:       wait;
228:    end process;
229:
230: END;
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5:
  6:
  7: library util;
  8: use util.pkg_util.all;
  9:
 10: library mctl;
 11: use mctl.pkg_mctl.all;
 12:
 13: library cctl;
 14: use cctl.pkg_ovm.all;
 15:
 16:
 17: package pkg_cctl is
 18:
 19:
 20:         type typ_cctl_cport_miso is record
 21:                 clk : std_logic;
 22:                 rd_en : std_logic;
 23:         end record;
 24:
 25:         constant init_cctl_cport_miso : typ_cctl_cport_miso := (
 26:                 clk => '0',
 27:                 rd_en => '0'
 28:         );
 29:
 30:
 31:         type typ_cctl_cport_mosi is record
 32:                 data : std_logic_vector(7 downto 0);
 33:                 empty : std_logic;
 34:                 full : std_logic;
 35:         end record;
 36:
 37:         constant init_cctl_cport_mosi : typ_cctl_cport_mosi := (
 38:                 data => (others => '0'),
 39:                 empty => '0',
 40:                 full => '0'
 41:         );
 42:
 43:
 44:
 45:         component cpt_cctl is
 46:
 47:                 port (
 48:                 i_clk : in std_logic;
 49:
 50:                         i_ovm0_video_miso : in typ_ovm_video_miso;
 51: --                      io_ovm0_sccb_bidir : inout typ_ovm_sccb_bidir;
 52: --                      o_ovm0_sccb_mosi : out typ_ovm_sccb_mosi;
 53:
 54:                         i_ovm1_video_miso : in typ_ovm_video_miso;
 55: --                      io_ovm1_sccb_bidir : inout typ_ovm_sccb_bidir;
 56: --                      o_ovm1_sccb_mosi : out typ_ovm_sccb_mosi;
 57:
 58:                         i_ovm2_video_miso : in typ_ovm_video_miso;
 59: --                      io_ovm2_sccb_bidir : inout typ_ovm_sccb_bidir;
 60: --                      o_ovm2_sccb_mosi : out typ_ovm_sccb_mosi;
 61:
 62:                         i_ovm3_video_miso : in typ_ovm_video_miso
 63: --                      io_ovm3_sccb_bidir : inout typ_ovm_sccb_bidir;
 64: --                      o_ovm3_sccb_mosi : out typ_ovm_sccb_mosi
 65:
 66:                 );
 67:
 68:         end component;
 69:
 70:
 71:         component cpt_cam is
 72:                 generic (
 73:                         ADDR : integer
 74:                 );
 75:                 port (
 76:                         i_clk : in std_logic;
 77:                         i_ovm_video_miso : in typ_ovm_video_miso
 78: --                      io_ovm_sccb_bidir : inout typ_ovm_sccb_bidir;
 79: --                      o_ovm_sccb_mosi : out typ_ovm_sccb_mosi
 80:                 );
 81:         end component;
 82:
 83:
 84:
 85:
 86:
 87:
 88:
 89:         component cpt_ovm is
 90:
 91:
 92:                 port (
 93:
 94:                         --i_clk : in std_logic;
 95:
 96:                         i_ovm_video_miso : in typ_ovm_video_miso;
 97:
 98:                         i_mport_mosi : in typ_mctl_mport_mosi;
 99:                         o_mport_miso : out typ_mctl_mport_miso
100:
```

```
101:                );
102:
103:        end component;
104:
105:
106:
107: end pkg_cctl;
108:
109:
110: package body pkg_cctl is
111: end pkg_cctl;
```

```
 1:
 2:
 3: library ieee;
 4: use ieee.std_logic_1164.all;
 5:
 6: --library cctl;
 7: --use cctl.pkg_cctl.all;
 8:
 9: package pkg_ovm is
10:
11:
12:         type typ_ovm_sccb_bidir is record
13:                 scl : std_logic;
14:                 sda : std_logic;
15:         end record;
16:
17:         constant init_ovm_sccb_bidir : typ_ovm_sccb_bidir := (
18:                 scl => 'Z',
19:                 sda => 'Z'
20:         );
21:
22:
23:         type typ_ovm_sccb_mosi is record
24:                 pwdn : std_logic;
25:                 xvclk : std_logic;
26:         end record;
27:
28:         constant init_ovm_sccb_mosi : typ_ovm_sccb_mosi := (
29:                 pwdn => '1',
30:                 xvclk => '0'
31:         );
32:
33:
34:         type typ_ovm_video_miso is record
35:                 pclk : std_logic;
36:                 data : std_logic_vector(7 downto 0);
37:                 href : std_logic;
38:                 vsync : std_logic;
39:         end record;
40:
41:         constant init_ovm_video_miso : typ_ovm_video_miso := (
42:                 pclk => '0',
43:                 data => (others => '0'),
44:                 href => '0',
45:                 vsync => '0'
46:         );
47:
48:
49: end pkg_ovm;
50:
51:
52: package body pkg_ovm is
53: end pkg_ovm;
```

```
  1:
  2: -- sim_ovm.vhd
  3: --
  4: -- Behavioural simulation of OVM7690 CameraCube.
  5: --
  6: -- Implemented:
  7: --              PLL (6MHz xvclk in, 24MHz pclk out)
  8: --              Internal registers
  9: --      Tristate ctrl/video bus
 10: --
 11: -- To be implemented:
 12: --      SCCB register read/write
 13: --              Video readout (RGB/YUV)
 14: --
 15: -- Won't be implemented
 16: --              Image processing
 17: --
 18:
 19:
 20: library ieee;
 21: use ieee.std_logic_1164.all;
 22: use ieee.std_logic_arith.all;
 23: use ieee.numeric_std.all;
 24:
 25: library unisim;
 26: use unisim.vcomponents.all;
 27:
 28: library util;
 29: use util.pkg_util.all;
 30:
 31: library cctl;
 32: use cctl.pkg_ovm.all;
 33:
 34:
 35: entity sim_ovm is
 36:         generic (
 37:                 SMALL_FRAME : string := "FALSE" -- Image size is reduced by a factor of ~100
 38:         );
 39:         port (
 40:                 i_ovm_sccb_mosi : in typ_ovm_sccb_mosi;
 41:                 io_ovm_sccb_bidir : inout typ_ovm_sccb_bidir;
 42:                 o_ovm_video_miso : out typ_ovm_video_miso
 43:         );
 44: end sim_ovm;
 45:
 46:
 47: architecture Behavioral of sim_ovm is
 48:
 49:         signal xvclk : std_logic := '0';
 50:
 51:         -- PLL
 52:         signal pll_clkfb : std_logic := '0';
 53:         signal pll_reset : std_logic := '1';
 54:         signal pll_lock : std_logic := '0';
 55:         signal pll_lock_n : std_logic := '1';
 56:
 57:         -- System
 58:         signal clk_8M : std_logic := '0';
 59:         signal clk_12M : std_logic := '0';
 60:         signal clk_24M : std_logic := '0';
 61:         signal clk_48M : std_logic := '0';
 62:         signal reset : std_logic := '1';
 63:
 64:
 65:         -- Register bank (refer to OV7690_CSP3 datasheet)
 66:         type typ_ovm_registers is array (0 to 255) of std_logic_vector(7 downto 0);
 67:
 68:         -- Register addresses
 69:         constant OVM_PIDH       : integer := 16#0A#; -- product ID MSB
 70:         constant OVM_PIDL       : integer := 16#0B#; -- product ID LSB
 71:         constant OVM_REG0C      : integer := 16#0C#; -- vflip,hmirror,BRswap,YUYVswap,busorder,tristate,overlay
 72:         constant OVM_REG0D      : integer := 16#0D#; -- VSstart,VSwidth
 73:         constant OVM_REG0E      : integer := 16#0E#; -- Sleep,Range,Drive
 74:         constant OVM_CLKRC      : integer := 16#11#; -- ExtClk,PreScale
 75:         constant OVM_REG12      : integer := 16#12#; -- Reset,Subsmp,ITU565,RAW,RGBfmt,OUTfmt
 76:         constant OVM_REG16      : integer := 16#16#; -- HsizeLSb,Voff,Hoff
 77:         constant OVM_HSIZE      : integer := 16#18#; -- HsizeMSB
 78:         constant OVM_VSIZE      : integer := 16#1A#; -- Vsize
 79:         constant OVM_MIDH       : integer := 16#1C#; -- mfr. ID MSB
 80:         constant OVM_MIDL       : integer := 16#1D#; -- mfr. ID LSB
 81:         constant OVM_REG28      : integer := 16#28#; -- DATAneg,HRtoHS,HSrev,HRrev,VSedge,VSneg
 82:         constant OVM_PLL                : integer := 16#29#; -- PLLdiv,PLLctl,PLLreset,YAVGsrc
 83:         constant OVM_REG3E      : integer := 16#3E#; -- PCLKgate,PCLKmult
 84:         constant OVM_REG3F      : integer := 16#3F#; -- PCLKrev
 85:         constant OVM_PWC0       : integer := 16#49#; -- DOVDD
 86:         constant OVM_REG62      : integer := 16#62#; -- TESTen,TESTmode
 87:
 88:
 89:         signal ovm_reg : typ_ovm_registers := (
 90:                 OVM_PIDH        => x"76",
 91:                 OVM_PIDL        => x"90",
 92:                 OVM_REG0C       => x"00",
 93:                 OVM_REG0D       => x"44",
 94:                 OVM_REG0E       => x"00",
 95:                 OVM_CLKRC       => x"00",
 96:                 OVM_REG12       => x"11",
 97:                 OVM_REG16       => x"08",
 98:                 OVM_HSIZE       => x"A0",
 99:                 OVM_VSIZE       => x"F0",
100:                 OVM_MIDH        => x"7F",
```

```
101:                OVM_MIDL        => x"A2",
102:                OVM_REG28       => x"00",
103:                OVM_PLL                 => x"A2",
104:                OVM_REG3E       => x"20",
105:                OVM_REG3F       => x"44",
106:                OVM_PWC0                => x"0D",
107:                OVM_REG62       => x"00",
108:                others => x"00"
109:        );
110:
111:
112:        -- Video generation
113:        signal red_pixel : integer := 0;
114:        signal green_pixel : integer := 0;
115:        signal blue_pixel : integer := 0;
116:
117:
118:
119:        function f(v1:integer; v2:integer) return integer is
120:                begin
121:                if (SMALL_FRAME = "FALSE") then
122:                  return v1;
123:                else
124:                  return v2;
125:                end if;
126:    end function;
127:
128:
129:
130:
131:
132:        -- ------------------------------------
133:        -- Characteristic timing constants
134:        -- ------------------------------------
135:
136:        constant H_ACTIVE_WIDTH : integer := f(640, 12);
137:        constant H_FRONTPORCH_WIDTH : integer := f(54, 2);
138:        constant H_SYNC_WIDTH : integer := f(16, 1);
139:        constant H_BACKPORCH_WIDTH : integer := f(70, 2);
140:
141:        constant V_ACTIVE_WIDTH : integer := f(480, 12);
142:        constant V_FRONTPORCH_WIDTH : integer := f(12, 1);
143:        constant V_SYNC_WIDTH : integer := f(4, 1);
144:        constant V_BACKPORCH_WIDTH : integer := f(16, 2);
145:
146:
147:        -- ------------------------------------------------------
148:        -- Derived timing constants (do not modify without reason)
149:        -- ------------------------------------------------------
150:
151:        constant H_ACTIVE_FIRST : integer := 0;
152:        constant H_ACTIVE_LAST : integer := H_ACTIVE_FIRST + H_ACTIVE_WIDTH - 1;
153:
154:        constant H_FRONTPORCH_FIRST : integer := H_ACTIVE_LAST + 1;
155:        constant H_FRONTPORCH_LAST : integer := H_FRONTPORCH_FIRST + H_FRONTPORCH_WIDTH - 1;
156:
157:        constant H_SYNC_FIRST : integer := H_FRONTPORCH_LAST + 1;
158:        constant H_SYNC_LAST : integer := H_SYNC_FIRST + H_SYNC_WIDTH - 1;
159:
160:        constant H_BACKPORCH_FIRST : integer := H_SYNC_LAST + 1;
161:        constant H_BACKPORCH_LAST : integer := H_BACKPORCH_FIRST + H_BACKPORCH_WIDTH - 1;
162:
163:        constant H_BLANK_FIRST : integer := H_FRONTPORCH_FIRST;
164:        constant H_BLANK_LAST : integer := H_BACKPORCH_LAST;
165:
166:        constant H_FRAME_FIRST : integer := H_ACTIVE_FIRST;
167:        constant H_FRAME_LAST : integer := H_BACKPORCH_LAST;
168:
169:        constant V_ACTIVE_FIRST : integer := 0;
170:        constant V_ACTIVE_LAST : integer := V_ACTIVE_FIRST + V_ACTIVE_WIDTH - 1;
171:
172:        constant V_FRONTPORCH_FIRST : integer := V_ACTIVE_LAST + 1;
173:        constant V_FRONTPORCH_LAST : integer := V_FRONTPORCH_FIRST + V_FRONTPORCH_WIDTH - 1;
174:
175:        constant V_SYNC_FIRST : integer := V_FRONTPORCH_LAST + 1;
176:        constant V_SYNC_LAST : integer := V_SYNC_FIRST + V_SYNC_WIDTH - 1;
177:
178:        constant V_BACKPORCH_FIRST : integer := V_SYNC_LAST + 1;
179:        constant V_BACKPORCH_LAST : integer := V_BACKPORCH_FIRST + V_BACKPORCH_WIDTH - 1;
180:
181:        constant V_BLANK_FIRST : integer := V_FRONTPORCH_FIRST;
182:        constant V_BLANK_LAST : integer := V_BACKPORCH_LAST;
183:
184:        constant V_FRAME_FIRST : integer := V_ACTIVE_FIRST;
185:        constant V_FRAME_LAST : integer := V_BACKPORCH_LAST;
186:
187:
188:        -- Video timing generation
189:        signal h_count : integer := 0;
190:        signal h_active : std_logic := '1';
191:        signal h_sync : std_logic := '0';
192:
193:        signal v_counter_enable : std_logic := '1';
194:        signal v_count : integer := 0;
195:        signal v_active : std_logic := '1';
196:        signal v_sync : std_logic := '0';
197:
198:        signal frame_counter_enable : std_logic := '1';
199:        signal frame_count : integer := 0;
200:        signal frame_active : std_logic := '1';
```

```
201:
202:
203:
204:
205:         signal subpixel : integer := 0;
206:         constant RGB : string := "RGBg";
207:         constant YCrCb : string := "YRyB";
208:
209:
210:
211:         -- Video output
212:         signal data : std_logic_vector(7 downto 0) := (others => '0');
213:         --signal pclk : std_logic := '0';
214:         --signal vsync : std_logic := '0';
215:         --signal href : std_logic := '0';
216:         signal tristate_ctrl : std_logic := '1';
217:         signal tristate_data : std_logic := '1';
218:
219:
220: begin
221:
222:         xvclk <= i_ovm_sccb_mosi.xvclk;
223:
224: --      xvclk_ibufg : IBUFG
225: --      port map (
226: --                      I => i_ovm_sccb_mosi.xvclk,
227: --                      O => xvclk
228: --      );
229:
230:
231:
232:
233:         -- 6 MHz ->
234:         --                      48 MHz
235:         --                      24 MHz -- pixel clock
236:         --                      12 MHz
237:         --                      8 MHz
238:
239:         pll_reset <= ovm_reg(OVM_PLL)(3);
240:
241:    ovm_pll_base : PLL_BASE
242:    generic map (
243:       BANDWIDTH => "OPTIMIZED", -- "HIGH", "LOW" or "OPTIMIZED"
244:       CLKFBOUT_MULT => 16, -- Multiply value for all CLKOUT clock outputs (1-64)
245:       CLKFBOUT_PHASE => 0.0, -- Phase offset in degrees of the clock feedback output (0.0-360.0).
246:       CLKIN_PERIOD => 166.667, -- 6 MHz     -- Input clock period in ns
247:       -- CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for CLKOUT# clock output (1-128)
248:       CLKOUT0_DIVIDE => 2,
249:       CLKOUT1_DIVIDE => 4,
250:       CLKOUT2_DIVIDE => 8,
251:       CLKOUT3_DIVIDE => 12,
252:       CLKOUT4_DIVIDE => 1,
253:       CLKOUT5_DIVIDE => 1,
254:       -- CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for CLKOUT# clock output (0.01-0.99).
255:       CLKOUT0_DUTY_CYCLE => 0.5,
256:       CLKOUT1_DUTY_CYCLE => 0.5,
257:       CLKOUT2_DUTY_CYCLE => 0.5,
258:       CLKOUT3_DUTY_CYCLE => 0.5,
259:       CLKOUT4_DUTY_CYCLE => 0.5,
260:       CLKOUT5_DUTY_CYCLE => 0.5,
261:       -- CLKOUT0_PHASE - CLKOUT5_PHASE: Output phase relationship for CLKOUT# clock output (-360.0-360.0).
262:       CLKOUT0_PHASE => 0.0,
263:       CLKOUT1_PHASE => 0.0,
264:       CLKOUT2_PHASE => 0.0,
265:       CLKOUT3_PHASE => 0.0,
266:       CLKOUT4_PHASE => 0.0,
267:       CLKOUT5_PHASE => 0.0,
268:       CLK_FEEDBACK => "CLKFBOUT", -- Clock source to drive CLKFBIN ("CLKFBOUT" or "CLKOUT0")
269:       COMPENSATION => "SYSTEM_SYNCHRONOUS", -- "SYSTEM_SYNCHRONOUS", "SOURCE_SYNCHRONOUS", "EXTERNAL"
270:       DIVCLK_DIVIDE => 1, -- Division value for all output clocks (1-52)
271:       REF_JITTER => 0.1, -- Reference Clock Jitter in UI (0.000-0.999).
272:       RESET_ON_LOSS_OF_LOCK => FALSE -- Must be set to FALSE
273:    )
274:    port map (
275:       CLKFBOUT => pll_clkfb, -- 1-bit output: PLL_BASE feedback output
276:       -- CLKOUT0 - CLKOUT5: 1-bit (each) output: Clock outputs
277:       CLKOUT0 => clk_48M,
278:       CLKOUT1 => clk_24M,
279:       CLKOUT2 => clk_12M,
280:       CLKOUT3 => clk_8M,
281:       CLKOUT4 => open,
282:       CLKOUT5 => open,
283:       LOCKED => pll_lock, -- 1-bit output: PLL_BASE lock status output
284:       CLKFBIN => pll_clkfb, -- 1-bit input: Feedback clock input
285:       CLKIN => xvclk, -- 1-bit input: Clock input
286:       RST => '0' -- 1-bit input: Reset input
287:    );
288:
289:
290:         pll_lock_n <= not pll_lock;
291:
292:
293:
294:
295:         -- System reset
296:         -- Released synchronously when PLL locks
297:         -- Reasserted asynchronously when PLL loses lock
298:         reset_fdp : fdp
299:         port map (
300:                 c => xvclk,
```

```
301:                d => pll_lock_n,
302:                q => reset,
303:                pre => pll_lock_n
304:        );
305:
306:
307:
308: --      pclk_clkout : cpt_clkout
309: --      generic map (
310: --              CLK_DIV2 => 0
311: --      )
312: --      port map (
313: --              i_clk => clk_24M,
314: --              o_clk => o_ovm_video_miso.pclk
315: --      );
316:
317:
318: --      process(clk_24M)
319: --      begin
320: --              if ( falling_edge(clk_24M) ) then
321: --                      red_pixel <= red_pixel + 1;
322: --                      green_pixel <= green_pixel + 1;
323: --                      blue_pixel <= blue_pixel + 1;
324: --              end if;
325: --      end process;
326:
327:
328: --      process(clk_24M)
329: --      begin
330: --
331: --      end process;
332:
333:
334:
335: -- -----------------------------------------------------
336: -- Video generation
337: -- -----------------------------------------------------
338:
339: --      subpixel <= (H_ACTIVE_WIDTH * v_count + h_count + (v_count mod 4)) mod 4
340: --              when h_active = '1' and v_active = '1'
341: --              else 0;
342:
343:         data <= std_logic_vector(to_unsigned(h_count mod 256, 8))
344:                 when h_active = '1' and v_active = '1'
345:                 else (others => '0');
346:
347: --      process(subpixel)
348: --      begin
349: --              data <= conv_std_logic_vector(character'pos(RGB(subpixel+1)), 8);
350: --      end process;
351:
352:
353:
354:
355:
356:
357: -- -----------------------------------------------------
358: -- Video timing generation
359: -- -----------------------------------------------------
360:
361:         h_counter : cpt_upcounter
362:         generic map (
363:                 INIT => 0
364:         )
365:         port map (
366:                 i_clk => clk_24M,
367:                 i_enable => '1',
368:                 i_lowest => H_FRAME_FIRST,
369:                 i_highest => H_FRAME_LAST,
370:                 i_increment => 1,
371:                 i_clear => reset,
372:                 i_preset => '0',
373:                 o_count => h_count,
374:                 o_carry => open
375:         );
376:
377:         h_active <= '1' when reset = '0' and h_count >= H_ACTIVE_FIRST and h_count <= H_ACTIVE_LAST else '0';
378:         h_sync <= '1' when reset = '0' and h_count >= H_SYNC_FIRST and h_count <= H_SYNC_LAST else '0';
379:
380:         v_counter_enable <= '1' when h_count = H_FRAME_LAST else '0';
381:
382:         v_counter : cpt_upcounter
383:         generic map (
384:                 INIT => 0
385:         )
386:         port map (
387:                 i_clk => clk_24M,
388:                 i_enable => v_counter_enable,
389:                 i_lowest => V_FRAME_FIRST,
390:                 i_highest => V_FRAME_LAST,
391:                 i_increment => 1,
392:                 i_clear => reset,
393:                 i_preset => '0',
394:                 o_count => v_count,
395:                 o_carry => open
396:         );
397:
398:         v_active <= '1' when reset = '0' and v_count >= V_ACTIVE_FIRST and v_count <= V_ACTIVE_LAST else '0';
399:         v_sync <= '1' when reset = '0' and v_count >= V_SYNC_FIRST and v_count <= V_SYNC_LAST else '0';
400:
```

```
401:
402:         frame_counter_enable <= '1' when h_count = H_FRAME_LAST and v_count = V_FRAME_LAST else '0';
403:
404:         frame_counter : cpt_upcounter
405:         generic map (
406:                 INIT => 0
407:         )
408:         port map (
409:                 i_clk => clk_24M,
410:                 i_enable => frame_counter_enable,
411:                 i_lowest => 0,
412:                 i_highest => 2**30-1,
413:                 i_increment => 1,
414:                 i_clear => reset,
415:                 i_preset => '0',
416:                 o_count => frame_count,
417:                 o_carry => open
418:         );
419:
420:         frame_active <= '1' when v_count >= 0 else '0';
421:
422:
423: -- -----------------------------------------------------
424: -- Output buffers
425: -- -----------------------------------------------------
426:
427:         tristate_data <= '0'; --not ovm_reg(OVM_REG0C)(2);
428:         tristate_ctrl <= '0'; --not ovm_reg(OVM_REG0C)(1);
429:
430:         o_ovm_video_miso.data <= data when tristate_data = '0' else (others => 'Z');
431:
432: --      gen_data_obuft :
433: --      for i in 0 to 7 generate
434: --              data_obuft : obuft
435: --              port map (
436: --                      i => data(i),
437: --                      o => o_ovm_video_miso.data(i),
438: --                      t => tristate_data
439: --              );
440: --      end generate;
441:
442:
443:         o_ovm_video_miso.pclk <= clk_24M when tristate_ctrl = '0' else 'Z';
444:
445: --      pclk_obuft : obuft
446: --      port map (
447: --              i => clk_24M,
448: --              o => o_ovm_video_miso.pclk,
449: --              t => tristate_ctrl
450: --      );
451:
452:
453:         o_ovm_video_miso.vsync <= v_sync when tristate_ctrl = '0' else 'Z';
454:
455: --      vsync_obuft : obuft
456: --      port map (
457: --              i => v_sync,
458: --              o => o_ovm_video_miso.vsync,
459: --              t => tristate_ctrl
460: --      );
461:
462:
463:         o_ovm_video_miso.href <= h_active when tristate_ctrl = '0' else 'Z';
464:
465: --      href_obuft : obuft
466: --      port map (
467: --              i => h_active,
468: --              o => o_ovm_video_miso.href,
469: --              t => tristate_ctrl
470: --      );
471:
472:
473:         io_ovm_sccb_bidir.sda <= 'Z';
474:
475: end Behavioral;
476:
```

```
    1:
    2: library ieee;
    3: use ieee.std_logic_1164.all;
    4:
    5: library unisim;
    6: use unisim.vcomponents.all;
    7:
    8:
    9: entity cpt_clkgen is
   10:         port (
   11:                 i_clk_24M : in std_logic;      -- 24 MHz input
   12:                 i_reset : in std_logic;
   13:                 o_clk_72M : out std_logic;     -- 72 MHz
   14:                 o_reset : out std_logic;
   15:                 o_async_reset : out std_logic;
   16:                 o_clk_288M : out std_logic;    -- 288 MHz
   17:                 o_clk_288M_180 : out std_logic;        -- 288 MHz
   18:                 o_mcb_drp_clk : out std_logic; -- 72 MHz
   19:                 o_clk_144M : out std_logic;    -- 144 MHz
   20:                 o_clk_144M_90 : out std_logic; -- 144 MHz
   21:                 o_pll_lock : out std_logic
   22:         );
   23: end cpt_clkgen;
   24:
   25: architecture cpt of cpt_clkgen is
   26:
   27: --   constant C_MEMCLK_PERIOD     : integer := 41666;
   28:    constant C_MEMCLK_PERIOD     : integer := 6944;
   29:
   30: --   constant C_RST_ACT_LOW      : integer := 0;
   31: --   constant C_INPUT_CLK_TYPE   : string  := "DIFFERENTIAL";
   32: --   constant C_CLKOUT0_DIVIDE   : integer := 2;
   33: --   constant C_CLKOUT1_DIVIDE   : integer := 2;
   34: --   constant C_CLKOUT2_DIVIDE   : integer := 8;
   35: --   constant C_CLKOUT3_DIVIDE   : integer := 8;
   36: --   constant C_CLKFBOUT_MULT   : integer := 4;
   37: --   constant C_DIVCLK_DIVIDE   : integer := 1
   38:
   39: --   constant C_INCLK_PERIOD     : integer := 2500;
   40:    constant C_RST_ACT_LOW      : integer := 0;
   41:    constant C_INPUT_CLK_TYPE   : string  := "SINGLE_ENDED";
   42:
   43:    constant C_CLKOUT0_DIVIDE       : integer := 2;
   44:    constant C_CLKOUT1_DIVIDE       : integer := 2;
   45:    constant C_CLKOUT2_DIVIDE       : integer := 8;
   46:    constant C_CLKOUT3_DIVIDE       : integer := 8;
   47:    constant C_CLKFBOUT_MULT        : integer := 24;
   48:    constant C_DIVCLK_DIVIDE        : integer := 1;
   49:    constant C_INCLK_PERIOD         : integer := ((C_MEMCLK_PERIOD * C_CLKFBOUT_MULT) / (C_DIVCLK_DIVIDE * C_CLKOUT0_DIVIDE * 2));
   50:
   51:
   52:
   53:
   54:   -- # of clock cycles to delay deassertion of reset. Needs to be a fairly
   55:   -- high number not so much for metastability protection, but to give time
   56:   -- for reset (i.e. stable clock cycles) to propagate through all state
   57:   -- machines and to all control signals (i.e. not all control signals have
   58:   -- resets, instead they rely on base state logic being reset, and the effect
   59:   -- of that reset propagating through the logic). Need this because we may not
   60:   -- be getting stable clock cycles while reset asserted (i.e. since reset
   61:   -- depends on PLL/DCM lock status)
   62:
   63:   constant RST_SYNC_NUM   : integer := 25;
   64:   constant CLK_PERIOD_NS  : real := (real(C_INCLK_PERIOD)) / 1000.0;
   65:   constant CLK_PERIOD_INT : integer := C_INCLK_PERIOD/1000;
   66:
   67:
   68:   signal   clk_2x_0            : std_logic;
   69:   signal   clk_2x_180          : std_logic;
   70:   signal   o_clk_72M_bufg          : std_logic;
   71:   signal   o_clk_72M_bufg_in       : std_logic;
   72:   signal   o_mcb_drp_clk_bufg_in : std_logic;
   73:   signal   clkfbout_clkfbin    : std_logic;
   74:   signal   rst_tmp            : std_logic;
   75:   signal   i_clk_24M_ibufg      : std_logic;
   76:   signal   sys_rst             : std_logic;
   77:   signal   o_reset_sync_r         : std_logic_vector(RST_SYNC_NUM-1 downto 0);
   78:   signal   powerup_o_pll_locked  : std_logic;
   79:   signal   syn_o_clk_72M_powerup_o_pll_locked : std_logic;
   80:   signal   locked              : std_logic;
   81:   signal   bufpll_mcb_locked   : std_logic;
   82:   signal   o_mcb_drp_clk_sig     : std_logic;
   83:
   84:   attribute max_fanout : string;
   85:   attribute syn_maxfan : integer;
   86:   attribute KEEP : string;
   87:   attribute max_fanout of o_reset_sync_r : signal is "10";
   88:   attribute syn_maxfan of o_reset_sync_r : signal is 10;
   89:   attribute KEEP of i_clk_24M_ibufg     : signal is "TRUE";
   90:
   91: begin
   92:
   93:   sys_rst  <= not(i_reset) when (C_RST_ACT_LOW /= 0) else i_reset;
   94:   o_clk_72M     <= o_clk_72M_bufg;
   95:   o_pll_lock <= bufpll_mcb_locked;
   96:   o_mcb_drp_clk <= o_mcb_drp_clk_sig;
   97:
   98: --  diff_input_clk : if(C_INPUT_CLK_TYPE = "DIFFERENTIAL") generate
   99: --      --***********************************************************************
  100: --      -- Differential input clock input buffers
```

```
101: --      --*************************************************************************
102: --      u_ibufg_i_clk_24M : IBUFGDS
103: --        generic map (
104: --          DIFF_TERM => TRUE
105: --        )
106: --        port map (
107: --          I  => i_clk_24M_p,
108: --          IB => i_clk_24M_n,
109: --          O  => i_clk_24M_ibufg
110: --          );
111: --  end generate;
112:
113:
114:   se_input_clk : if(C_INPUT_CLK_TYPE = "SINGLE_ENDED") generate
115:       --*************************************************************************
116:       -- SINGLE_ENDED input clock input buffers
117:       --*************************************************************************
118:       u_ibufg_i_clk_24M : IBUFG
119:         port map (
120:           I  => i_clk_24M,
121:           O  => i_clk_24M_ibufg
122:           );
123:   end generate;
124:
125:   --*************************************************************************
126:   -- Global clock generation and distribution
127:   --*************************************************************************
128:
129:     u_pll_adv : PLL_ADV
130:     generic map
131:         (
132:          BANDWIDTH          => "OPTIMIZED",
133:          CLKIN1_PERIOD      => CLK_PERIOD_NS,
134:          CLKIN2_PERIOD      => CLK_PERIOD_NS,
135:          CLKOUT0_DIVIDE     => C_CLKOUT0_DIVIDE,
136:          CLKOUT1_DIVIDE     => C_CLKOUT1_DIVIDE,
137:          CLKOUT2_DIVIDE     => C_CLKOUT2_DIVIDE,
138:          CLKOUT3_DIVIDE     => C_CLKOUT3_DIVIDE,
139:          CLKOUT4_DIVIDE     => 1,
140:          CLKOUT5_DIVIDE     => 1,
141:          CLKOUT0_PHASE      => 0.000,
142:          CLKOUT1_PHASE      => 180.000,
143:          CLKOUT2_PHASE      => 0.000,
144:          CLKOUT3_PHASE      => 0.000,
145:          CLKOUT4_PHASE      => 0.000,
146:          CLKOUT5_PHASE      => 0.000,
147:          CLKOUT0_DUTY_CYCLE => 0.500,
148:          CLKOUT1_DUTY_CYCLE => 0.500,
149:          CLKOUT2_DUTY_CYCLE => 0.500,
150:          CLKOUT3_DUTY_CYCLE => 0.500,
151:          CLKOUT4_DUTY_CYCLE => 0.500,
152:          CLKOUT5_DUTY_CYCLE => 0.500,
153:          SIM_DEVICE         => "SPARTAN6",
154:          COMPENSATION       => "INTERNAL",
155:          DIVCLK_DIVIDE      => C_DIVCLK_DIVIDE,
156:          CLKFBOUT_MULT      => C_CLKFBOUT_MULT,
157:          CLKFBOUT_PHASE     => 0.0,
158:          REF_JITTER         => 0.005000
159:          )
160:         port map
161:           (
162:            CLKFBIN          => clkfbout_clkfbin,
163:            CLKINSEL         => '1',
164:            CLKIN1           => i_clk_24M_ibufg,
165:            CLKIN2           => '0',
166:            DADDR            => (others => '0'),
167:            DCLK             => '0',
168:            DEN              => '0',
169:            DI               => (others => '0'),
170:            DWE              => '0',
171:            REL              => '0',
172:            RST              => sys_rst,
173:            CLKFBDCM         => open,
174:            CLKFBOUT         => clkfbout_clkfbin,
175:            CLKOUTDCM0       => open,
176:            CLKOUTDCM1       => open,
177:            CLKOUTDCM2       => open,
178:            CLKOUTDCM3       => open,
179:            CLKOUTDCM4       => open,
180:            CLKOUTDCM5       => open,
181:            CLKOUT0          => clk_2x_0,
182:            CLKOUT1          => clk_2x_180,
183:            CLKOUT2          => o_clk_72M_bufg_in,
184:            CLKOUT3          => o_mcb_drp_clk_bufg_in,
185:            CLKOUT4          => open,
186:            CLKOUT5          => open,
187:            DO               => open,
188:            DRDY             => open,
189:            LOCKED           => locked
190:            );
191:
192:     U_BUFG_o_clk_72M : BUFG
193:     port map
194:     (
195:      O => o_clk_72M_bufg,
196:      I => o_clk_72M_bufg_in
197:      );
198:
199:     --U_BUFG_CLK1 : BUFG
200:     -- port map (
```

```
201:    --  O => o_mcb_drp_clk_sig,
202:    --  I => o_mcb_drp_clk_bufg_in
203:    --  );
204:
205:    U_BUFG_CLK1 : BUFGCE
206:     port map (
207:      O => o_mcb_drp_clk_sig,
208:      I => o_mcb_drp_clk_bufg_in,
209:      CE => locked
210:      );
211:
212:    process (o_mcb_drp_clk_sig, sys_rst)
213:    begin
214:       if(sys_rst = '1') then
215:          powerup_o_pll_locked <= '0';
216:       elsif (o_mcb_drp_clk_sig'event and o_mcb_drp_clk_sig = '1') then
217:          if (bufpll_mcb_locked = '1') then
218:             powerup_o_pll_locked <= '1';
219:          end if;
220:       end if;
221:    end process;
222:
223:
224:    process (o_clk_72M_bufg, sys_rst)
225:    begin
226:       if(sys_rst = '1') then
227:          syn_o_clk_72M_powerup_o_pll_locked <= '0';
228:       elsif (o_clk_72M_bufg'event and o_clk_72M_bufg = '1') then
229:          if (bufpll_mcb_locked = '1') then
230:             syn_o_clk_72M_powerup_o_pll_locked <= '1';
231:          end if;
232:       end if;
233:    end process;
234:
235:
236:    --***************************************************************************
237:    -- Reset synchronization
238:    -- NOTES:
239:    --   1. shut down the whole operation if the PLL hasn't yet locked (and
240:    --      by inference, this means that external sys_rst has been asserted -
241:    --      PLL deasserts LOCKED as soon as sys_rst asserted)
242:    --   2. asynchronously assert reset. This was we can assert reset even if
243:    --      there is no clock (needed for things like 3-stating output buffers).
244:    --      reset deassertion is synchronous.
245:    --   3. asynchronous reset only look at o_pll_lock from PLL during power up. After
246:    --      power up and o_pll_lock is asserted, the powerup_o_pll_locked will be asserted
247:    --      forever until sys_rst is asserted again. PLL will lose lock when FPGA
248:    --      enters suspend mode. We don't want reset to MCB get
249:    --      asserted in the application that needs suspend feature.
250:    --***************************************************************************
251:
252:
253:   o_async_reset <= sys_rst or not(powerup_o_pll_locked);
254:   -- o_async_reset <= rst_tmp;
255:   rst_tmp <= sys_rst or not(syn_o_clk_72M_powerup_o_pll_locked);
256:   -- rst_tmp <= sys_rst or not(powerup_o_pll_locked);
257:
258: process (o_clk_72M_bufg, rst_tmp)
259:   begin
260:     if (rst_tmp = '1') then
261:       o_reset_sync_r <= (others => '1');
262:     elsif (rising_edge(o_clk_72M_bufg)) then
263:       o_reset_sync_r <= o_reset_sync_r(RST_SYNC_NUM-2 downto 0) & '0';  -- logical left shift by one (pads with 0)
264:     end if;
265:   end process;
266:
267:   o_reset    <= o_reset_sync_r(RST_SYNC_NUM-1);
268:
269:
270: BUFPLL_MCB_INST : BUFPLL_MCB
271: port map
272: ( IOCLK0         => o_clk_288M,
273:   IOCLK1         => o_clk_288M_180,
274:   LOCKED         => locked,
275:   GCLK           => o_mcb_drp_clk_sig,
276:   SERDESSTROBE0  => o_clk_144M,
277:   SERDESSTROBE1  => o_clk_144M_90,
278:   PLLIN0         => clk_2x_0,
279:   PLLIN1         => clk_2x_180,
280:   LOCK           => bufpll_mcb_locked
281:   );
282:
283: end cpt;
284:
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5: library cctl;
  6: use cctl.pkg_ovm.all;
  7:
  8: library mcu;
  9: use mcu.pkg_mcu.all;
 10:
 11: library util;
 12: use util.pkg_util.all;
 13:
 14:
 15: entity cpt_iobus_i2c is
 16:
 17:         generic (
 18:                 DEVICE_ID : std_logic_vector(31 downto 0);
 19:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0)
 20:         );
 21:
 22:         port (
 23:
 24:                 i_clk : in std_logic;
 25:                 i_enable : in std_logic;
 26:
 27:                 i_iobus_mosi : in typ_mcu_iobus_mosi;
 28:                 o_iobus_miso : out typ_mcu_iobus_miso;
 29:
 30:                 i_scl_clk_div : in integer;
 31:
 32:                 io_scl : inout std_logic;
 33:                 io_sda : inout std_logic
 34:
 35:         );
 36:
 37: end cpt_iobus_i2c;
 38:
 39: architecture Behavioral of cpt_iobus_i2c is
 40:
 41:
 42:
 43:         component cpt_i2c is
 44:
 45:                 port (
 46:
 47:                         i_clk : in std_logic;
 48:                         i_enable : in std_logic;
 49:
 50:                         i_scl_clk_div : in integer;
 51:
 52:                         i_addr : in std_logic_vector(6 downto 0);
 53:
 54:                         o_rd_data : out std_logic_vector(7 downto 0);
 55:                         o_rd_data_strobe : out std_logic;
 56:                         i_rd_start : in std_logic;
 57:                         o_rd_done : out std_logic;
 58:
 59:                         i_wr_data_available : in std_logic;
 60:                         i_wr_data : in std_logic_vector(7 downto 0);
 61:                         o_wr_data_strobe : out std_logic;
 62:                         i_wr_start : in std_logic;
 63:                         o_wr_done : out std_logic;
 64:
 65:                         io_i2c_scl : inout std_logic;
 66:                         io_i2c_sda : inout std_logic
 67:
 68:                 );
 69:
 70:         end component;
 71:
 72:
 73:
 74:
 75:
 76:         constant STATE_IDLE : integer := 16#00#;
 77:         constant STATE_WRITE : integer := 16#10#;
 78:         constant STATE_READ : integer := 16#20#;
 79:         constant STATE_READY : integer := 16#30#;
 80:
 81:         signal state : integer := STATE_IDLE;
 82:
 83:
 84:         signal dev_addr : std_logic_vector(6 downto 0);
 85:         signal reg_addr : std_logic_vector(7 downto 0);
 86:         signal reg_data : std_logic_vector(7 downto 0);
 87:
 88:         signal read_data : std_logic_vector(7 downto 0);
 89:         signal read_data_strobe : std_logic;
 90:         signal read_start : std_logic := '0';
 91:         signal read_done : std_logic;
 92:
 93:         signal write_data_available : std_logic := '0';
 94:         signal write_data : std_logic_vector(7 downto 0) := x"55";
 95:         signal write_data_strobe : std_logic;
 96:         signal write_start : std_logic := '0';
 97:         signal write_done : std_logic;
 98:
 99:
100:
```

```
101: begin
102:
103:
104:
105:
106:
107:         i2c : cpt_i2c
108:         port map (
109:                 i_clk => i_clk,
110:                 i_enable => i_enable,
111:                 i_scl_clk_div => i_scl_clk_div,
112:                 i_addr => dev_addr,
113:                 o_rd_data => read_data,
114:                 o_rd_data_strobe => read_data_strobe,
115:                 i_rd_start => read_start,
116:                 o_rd_done => read_done,
117:                 i_wr_data_available => write_data_available,
118:                 i_wr_data => write_data,
119:                 o_wr_data_strobe => write_data_strobe,
120:                 i_wr_start => write_start,
121:                 o_wr_done => write_done,
122:                 io_i2c_scl => io_scl,
123:                 io_i2c_sda => io_sda
124:         );
125:
126:
127:         process(i_clk)
128:         begin
129:                 if ( rising_edge(i_clk) and read_data_strobe = '1' ) then
130:                         o_iobus_miso.read_data <= x"000000" & read_data;
131:                 end if;
132:         end process;
133:
134:
135:
136:
137:         process(i_clk)
138:         begin
139:                 if ( rising_edge(i_clk) ) then
140:                         case state is
141:
142:                                 when STATE_IDLE+0 =>
143:                                         if ( i_iobus_mosi.addr_strobe = '1' ) then
144:                                                 dev_addr <= "1000011"; -- OVM camera device ID
145:                                                 reg_addr <= i_iobus_mosi.address(7 downto 0);
146:                                         end if;
147:                                         if ( i_iobus_mosi.write_strobe = '1' ) then
148:                                                 reg_data <= i_iobus_mosi.write_data(7 downto 0);
149:                                                 state <= STATE_WRITE;
150:                                         end if;
151:                                         if ( i_iobus_mosi.read_strobe = '1' ) then
152:                                                 state <= STATE_READ;
153:                                         end if;
154:
155:                                 when STATE_WRITE+0 =>
156:                                         write_start <= '0';
157:                                         if ( write_done = '1' ) then
158:                                                 state <= state + 1;
159:                                         end if;
160:                                 when STATE_WRITE+1 =>
161:                                         write_start <= '1';
162:                                         write_data <= reg_addr;
163:                                         write_data_available <= '1';
164:                                         if ( write_done = '0' ) then
165:                                                 state <= state + 1;
166:                                         end if;
167:                                 when STATE_WRITE+2 =>
168:                                         write_start <= '0';
169:                                         if ( write_data_strobe = '1' ) then
170:                                                 state <= state + 1;
171:                                         end if;
172:                                 when STATE_WRITE+3 =>
173:                                         write_data <= reg_data;
174:                                         write_data_available <= '1';
175:                                         if ( write_data_strobe = '1' ) then
176:                                                 state <= state + 1;
177:                                         end if;
178:                                 when STATE_WRITE+4 =>
179:                                         write_data_available <= '0';
180:                                         if ( write_done = '1' ) then
181:                                                 state <= STATE_READY;
182:                                         end if;
183:
184:                                 when STATE_READ+0 =>
185:                                         read_start <= '0';
186:                                         if ( read_done = '1' ) then
187:                                                 state <= state + 1;
188:                                         end if;
189:                                 when STATE_READ+1 =>
190:                                         read_start <= '1';
191:                                         if ( read_done = '0' ) then
192:                                                 state <= state + 1;
193:                                         end if;
194:                                 when STATE_READ+2 =>
195:                                         read_start <= '0';
196:                                         if ( read_done = '1' ) then
197:                                                 state <= STATE_READY;
198:                                         end if;
199:
200:                                 when STATE_READY+0 =>
```

```
                                        o_iobus_miso.ready <= '1';
                                        state <= state + 1;
                                when STATE_READY+1 =>
                                        o_iobus_miso.ready <= '0';
                                        state <= STATE_IDLE;

                                when others =>
                                        state <= STATE_IDLE;

                        end case;
                end if;
        end process;









        --o_clk => o_sccb_mosi.scl



--      sccb_scl_gate : cpt_clk_gate
--      port map (
--              i_clk => i_clk,
--              i_enable => i_enable,
--              i_div => i_scl_div,
--              o_clk_pgate => scl_pgate,
--              o_clk_ngate => scl_ngate
--      );



--
--      sccb_write_counter : cpt_upcounter
----    generic map (
----            INIT => 0
----    );
--      port map (
--              i_clk => i_clk,
--              i_enable => scl_pgate,
--              i_lowest => 0,
--              i_highest => 9,
--              i_increment => 1,
--              i_clear => sccb_done,
--              o_count => open,
--              o_carry => sccb_write_done
--      );
--
--
--
--
--      sccb_read_upcounter : cpt_upcounter
----    generic map (
----            INIT => 0
----    );
--      port map (
--              i_clk => i_clk,
--              i_enable => scl_pgate,
--              i_lowest => 0,
--              i_highest => 9,
--              i_increment => 1,
--              i_clear => sccb_done,
--              o_count => open,
--              o_carry => sccb_read_done
--      );
--
--
--
--
----    process(i_clk)
----    begin
----            if ( rising_edge(i_clk) ) then
----                    o_mcu_o_iobus_miso.ready <= i_mcu_i_iobus_mosi.addr_strobe;
----            end if;
----    end process;
--
----    iobus_i2c_io : cpt_iobus_i2c_io
----    port map (
----            i_clk <= i_clk,
----            i_start_ip <= sccb_start_ip,
----            i_start_addr <= sccb_start_addr,
----            i_start_write <= sccb_start_write,
----            i_start_read <= sccb_start_read,
----
----            io_sccb_bidir <= io_sccb_bidir,
----            o_sccb_mosi <= o_sccb_mosi
----
----    )
--
--
--
--      process(i_clk)
```

```
301: --      begin
302: --              if ( rising_edge(i_clk) ) then
303: --
304: --                      sccb_ip_start <= '0';
305: --                      sccb_addr_start <= '0';
306: --                      sccb_write_start <= '0';
307: --                      sccb_read_start <= '0';
308: --
309: --                      case sccb_state is
310: --
311: --                              when SCCB_STATE_IDLE =>
312: --                                      if ( i_i_iobus_mosi.addr_strobe = '1' and (i_i_iobus_mosi.address and DEVICE_ID_MASK) = DEVICE_ID ) then
313: --                                              ip_addr <= SCCB_BUS_ADDR(7 downto 1) & i_i_iobus_mosi.read_strobe;
314: --                                              sub_addr <= i_i_iobus_mosi.address(9 downto 2);
315: --                                              write_data <= i_i_iobus_mosi.write_data(7 downto 0);
316: --                                              if ( i_i_iobus_mosi.read_strobe = '1' ) then
317: --                                                      sccb_state <= SCCB_STATE_READ;
318: --                                              elsif ( i_i_iobus_mosi.write_strobe = '1' ) then

319: --                                                      sccb_state <= SCCB_STATE_WRITE;
320: --                                              end if;
321: --                                      end if;
322: --
323: --
324: --                              when SCCB_STATE_WRITE+0 =>
325: --                                      if ( sccb_done = '1' ) then
326: --                                              sccb_ip_start <= '1';
327: --                                              sccb_state <= sccb_state + 1;
328: --                                      end if;
329: --
330: --                              when SCCB_STATE_WRITE+1 =>
331: --                                      if ( sccb_ip_done = '1' ) then
332: --                                              sccb_addr_start <= '1';
333: --                                              sccb_state <= sccb_state + 1;
334: --                                      end if;
335: --
336: --                              when SCCB_STATE_WRITE+2 =>
337: --                                      if ( sccb_addr_done = '1' ) then
338: --                                              sccb_write_start <= '1';
339: --                                              sccb_state <= SCCB_STATE_IDLE;
340: --                                      end if;
341: --
342: --                              when SCCB_STATE_WRITE+3 =>
343: --                                      if ( sccb_write_done = '1' ) then
344: --                                              sccb_state <= SCCB_STATE_IDLE;
345: --                                      end if;
346: --
347: --
348: --                              when SCCB_STATE_READ+0 =>
349: --                                      if ( sccb_done = '1' ) then
350: --                                              sccb_ip_start <= '1';
351: --                                              sccb_state <= sccb_state + 1;
352: --                                      end if;
353: --
354: --                              when SCCB_STATE_READ+1 =>
355: --                                      if ( sccb_ip_done = '1' ) then
356: --                                              sccb_addr_start <= '1';
357: --                                              sccb_state <= sccb_state + 1;
358: --                                      end if;
359: --
360: --                              when SCCB_STATE_READ+2 =>
361: --                                      if ( sccb_addr_done = '1' ) then
362: --                                              sccb_read_start <= '1';
363: --                                              sccb_state <= sccb_state + 1;
364: --                                      end if;
365: --
366: --                              when SCCB_STATE_READ+3 =>
367: --                                      if ( sccb_read_done = '1' ) then
368: --                                              sccb_state <= SCCB_STATE_IDLE;
369: --                                      end if;
370: --
371: --                              when others =>
372: --                                      sccb_state <= SCCB_STATE_IDLE;
373: --
374: --                      end case;
375: --              end if;
376: --      end process;
377: --
378: --
379: --      process(i_clk)
380: --      begin
381: --              if ( rising_edge(i_clk) ) then
382: --
383: --                      o_o_iobus_miso.read_data <= (others => '0');
384: --                      o_o_iobus_miso.ready <= '0';
385: --
386: --                      case sccb_io_state is
387: --
388: --                              when SCCB_IO_STATE_IDLE+0 =>
389: --                                      sccb_done <= '1';
390: --                                      sccb_ip_done <= '0';
391: --                                      sccb_addr_done <= '0';
392: --                                      sccb_write_done <= '0';
393: --                                      sccb_read_done <= '0';
394: --                                      if ( sccb_ip_start = '1' ) then
395: --                                              sccb_done <= '0';
396: --                                              sccb_write_data <= ip_addr & 'Z';
397: --                                              sccb_io_state <= SCCB_IO_STATE_WRITE;
398: --                                              --sccb_io_state <= SCCB_IO_STATE_IP;
399: --                                      end if;
```

```
400: --                                         if ( sccb_addr_start = '1' ) then
401: --                                                 sccb_done <= '0';
402: --                                                 sccb_write_data <= sub_addr & 'Z';
403: --                                                 sccb_io_state <= SCCB_IO_STATE_WRITE;
404: --                                                 --sccb_io_state <= SCCB_IO_STATE_ADDR;
405: --                                         end if;
406: --                                         if ( sccb_write_start = '1' ) then
407: --                                                 sccb_done <= '0';
408: --                                                 sccb_write_data <= write_data & 'Z';
409: --                                                 sccb_io_state <= SCCB_IO_STATE_WRITE;
410: --                                         end if;
411: --                                         if ( sccb_read_start = '1' ) then
412: --                                                 sccb_done <= '0';
413: --                                                 sccb_io_state <= SCCB_IO_STATE_READ;
414: --                                         end if;
415: --
416: --
417: --                                   when SCCB_IO_STATE_WRITE+0 =>
418: --                                         sccb_write_enable <= '1';
419: --                                         sccb_io_state <= sccb_io_state + 1;
420: --
421: --                                   when SCCB_IO_STATE_WRITE+1 =>
422: --                                         if ( sccb_write_done = '1' ) then
423: --                                                 sccb_io_state <= sccb_io_state + 1;
424: --                                         else
425: --                                                 io_sccb_bidir.sda <= sccb_write_data(0);
426: --                                                 sccb_write_data <= '0' & sccb_write_data(8 downto 1);
427: --                                         end if;
428: --
429: --                                   when SCCB_IO_STATE_WRITE+2 =>
430: --                                         o_o_iobus_miso.ready <= '1';
431: --                                         sccb_io_state <= SCCB_IO_STATE_DONE;
432: --
433: --
434: --                                   when SCCB_IO_STATE_DONE+0 =>
435: --                                         sccb_ip_done <= '1';
436: --                                         sccb_addr_done <= '1';
437: --                                         sccb_write_done <= '1';
438: --                                         sccb_read_done <= '1';
439: --                                         sccb_io_state <= SCCB_IO_STATE_IDLE;
440: --
441: --
442: --
443: --                                   when others =>
444: --                                         null;
445: --
446: --                             end case;
447: --
448: --
449: --                     end if;
450: --             end process;
451: --
452:
453:
454:
455:
456:
457:
458:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
470:
471:
472:
473: end Behavioral;
474:
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5: library cctl;
  6: use cctl.pkg_ovm.all;
  7:
  8: library mcu;
  9: use mcu.pkg_mcu.all;
 10:
 11: library util;
 12: use util.pkg_util.all;
 13:
 14:
 15: entity cpt_iobus_sccb is
 16:
 17:         generic (
 18:                 DEVICE_ID : std_logic_vector(31 downto 0);
 19:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0)
 20:         );
 21:
 22:         port (
 23:
 24:                 i_clk : in std_logic;
 25:                 i_enable : in std_logic;
 26:
 27:                 i_iobus_mosi : in typ_mcu_iobus_mosi;
 28:                 o_iobus_miso : out typ_mcu_iobus_miso;
 29:
 30:                 i_scl_clk_div : in integer;
 31:                 i_dev_addr : in std_logic_vector(6 downto 0);
 32:
 33:                 io_scl : inout std_logic;
 34:                 io_sda : inout std_logic
 35:
 36:         );
 37:
 38: end cpt_iobus_sccb;
 39:
 40: architecture Behavioral of cpt_iobus_sccb is
 41:
 42:
 43:
 44:         component cpt_i2c is
 45:
 46:                 port (
 47:
 48:                         i_clk : in std_logic;
 49:                         i_enable : in std_logic;
 50:
 51:                         i_scl_clk_div : in integer;
 52:
 53:                         i_addr : in std_logic_vector(6 downto 0);
 54:
 55:                         o_rd_data : out std_logic_vector(7 downto 0);
 56:                         o_rd_data_strobe : out std_logic;
 57:                         i_rd_start : in std_logic;
 58:                         o_rd_done : out std_logic;
 59:
 60:                         i_wr_data_available : in std_logic;
 61:                         i_wr_data : in std_logic_vector(7 downto 0);
 62:                         o_wr_data_strobe : out std_logic;
 63:                         i_wr_start : in std_logic;
 64:                         o_wr_done : out std_logic;
 65:
 66:                         io_i2c_scl : inout std_logic;
 67:                         io_i2c_sda : inout std_logic
 68:
 69:                 );
 70:
 71:         end component;
 72:
 73:
 74:
 75:
 76:         constant STATE_IDLE : integer := 16#00#;
 77:         constant STATE_WRITE : integer := 16#10#;
 78:         constant STATE_READ : integer := 16#20#;
 79:         constant STATE_READY : integer := 16#30#;
 80:
 81:         signal state : integer := STATE_IDLE;
 82:
 83:
 84:         --signal dev_addr : std_logic_vector(6 downto 0) := "0010001"; -- Read:0x42, Write:0x43, 7bit addr: 0x21
 85:         signal reg_addr : std_logic_vector(7 downto 0);
 86:         signal reg_data : std_logic_vector(7 downto 0);
 87:
 88:         signal read_data : std_logic_vector(7 downto 0);
 89:         signal read_data_strobe : std_logic;
 90:         signal read_start : std_logic := '0';
 91:         signal read_done : std_logic;
 92:
 93:         signal write_data_available : std_logic := '0';
 94:         signal write_data : std_logic_vector(7 downto 0) := x"55";
 95:         signal write_data_strobe : std_logic;
 96:         signal write_start : std_logic := '0';
 97:         signal write_done : std_logic;
 98:
 99:
100:
```

```
101: begin
102:
103:
104:
105:
106:        i2c : cpt_i2c
107:        port map (
108:                i_clk => i_clk,
109:                i_enable => i_enable,
110:                i_scl_clk_div => i_scl_clk_div,
111:                i_addr => i_dev_addr,
112:                o_rd_data => read_data,
113:                o_rd_data_strobe => read_data_strobe,
114:                i_rd_start => read_start,
115:                o_rd_done => read_done,
116:                i_wr_data_available => write_data_available,
117:                i_wr_data => write_data,
118:                o_wr_data_strobe => write_data_strobe,
119:                i_wr_start => write_start,
120:                o_wr_done => write_done,
121:                io_i2c_scl => io_scl,
122:                io_i2c_sda => io_sda
123:        );
124:
125:
126:        process(i_clk)
127:        begin
128:                if ( rising_edge(i_clk) and read_data_strobe = '1' ) then
129:                        o_iobus_miso.read_data <= x"000000" & read_data;
130:                end if;
131:        end process;
132:
133:
134:
135:
136:        process(i_clk)
137:        begin
138:                if ( rising_edge(i_clk) ) then
139:                        case state is
140:
141:                                when STATE_IDLE+0 =>
142:
143:                                        if ( (i_iobus_mosi.address and DEVICE_ID_MASK) = (DEVICE_ID and DEVICE_ID_MASK) ) then

144:                                                if ( i_iobus_mosi.addr_strobe = '1' ) then
145:                                                        reg_addr <= i_iobus_mosi.address(9 downto 2);
146:                                                end if;
147:                                                if ( i_iobus_mosi.write_strobe = '1' ) then
148:                                                        reg_data <= i_iobus_mosi.write_data(7 downto 0);
149:                                                        state <= STATE_WRITE;
150:                                                end if;
151:                                                if ( i_iobus_mosi.read_strobe = '1' ) then
152:                                                        state <= STATE_READ;
153:                                                end if;
154:                                        end if;
155:
156:                                when STATE_WRITE+0 =>
157:                                        write_start <= '0';
158:                                        if ( write_done = '1' ) then
159:                                                state <= state + 1;
160:                                        end if;
161:                                when STATE_WRITE+1 =>
162:                                        write_start <= '1';
163:                                        write_data <= reg_addr;
164:                                        write_data_available <= '1';
165:                                        if ( write_done = '0' ) then
166:                                                state <= state + 1;
167:                                        end if;
168:                                when STATE_WRITE+2 =>
169:                                        write_start <= '0';
170:                                        if ( write_data_strobe = '1' ) then
171:                                                state <= state + 1;
172:                                        end if;
173:                                when STATE_WRITE+3 =>
174:                                        write_data <= reg_data;
175:                                        write_data_available <= '1';
176:                                        if ( write_data_strobe = '1' ) then
177:                                                state <= state + 1;
178:                                        end if;
179:                                when STATE_WRITE+4 =>
180:                                        write_data_available <= '0';
181:                                        if ( write_done = '1' ) then
182:                                                state <= STATE_READY;
183:                                        end if;
184:
185:
186:
187:
188:                                when STATE_READ+0 =>
189:                                        write_start <= '0';
190:                                        if ( write_done = '1' ) then
191:                                                state <= state + 1;
192:                                        end if;
193:                                when STATE_READ+1 =>
194:                                        write_start <= '1';
195:                                        write_data <= reg_addr;
196:                                        write_data_available <= '1';
197:                                        if ( write_done = '0' ) then
198:                                                state <= state + 1;
199:                                        end if;
```

```
200:                                when STATE_READ+2 =>
201:                                    write_start <= '0';
202:                                    if ( write_data_strobe = '1' ) then
203:                                            state <= state + 1;
204:                                    end if;
205:                                when STATE_READ+3 =>
206:                                    write_data_available <= '0';
207:                                    if ( write_done = '1' ) then
208:                                            state <= state + 1;
209:                                    end if;
210:                                when STATE_READ+4 =>
211:                                    read_start <= '0';
212:                                    if ( read_done = '1' ) then
213:                                            state <= state + 1;
214:                                    end if;
215:                                when STATE_READ+5 =>
216:                                    read_start <= '1';
217:                                    if ( read_done = '0' ) then
218:                                            state <= state + 1;
219:                                    end if;
220:                                when STATE_READ+6 =>
221:                                    read_start <= '0';
222:                                    if ( read_done = '1' ) then
223:                                            state <= STATE_READY;
224:                                    end if;
225:
226:                                when STATE_READY+0 =>
227:                                    o_iobus_miso.ready <= '1';
228:                                    state <= state + 1;
229:                                when STATE_READY+1 =>
230:                                    o_iobus_miso.ready <= '0';
231:                                    state <= STATE_IDLE;
232:
233:                                when others =>
234:                                    state <= STATE_IDLE;
235:
236:                        end case;
237:                end if;
238:        end process;
239:
240:
241: end Behavioral;
242:
```

```
 1:
 2: -- util/cpt_upcounter.vhd
 3:
 4: -- Produces an integer o_count
 5: -- that increases from i_lowest to i_highest
 6: -- in steps of i_increment
 7: -- when i_enable is high
 8: -- during a rising edge of i_clk
 9:
10:
11: library ieee;
12: use ieee.std_logic_1164.all;
13:
14: --library util;
15: --use util.pkg_util.ALL;
16:
17:
18: entity cpt_upcounter_testing is
19:         generic (
20:                 INIT : integer := -1
21:         );
22:         port (
23:                 i_clk : in std_logic;
24:                 i_enable : in std_logic;
25:                 i_lowest : in integer;
26:                 i_highest : in integer;
27:                 i_increment : in integer;
28:                 i_clear : in std_logic;
29:                 i_preset : in std_logic;
30:                 o_count : out integer := INIT;
31:                 o_carry : out std_logic
32:         );
33: end cpt_upcounter_testing;
34:
35:
36: architecture Behavioral of cpt_upcounter_testing is
37:
38: signal count : integer := INIT;
39: signal lowest : integer := 0;
40: signal highest : integer := 0;
41: signal increment : integer := 1;
42: signal reset : std_logic := '1';
43: signal carry : std_logic := '0';
44:
45: begin
46:
47:         o_count <= count;
48:         o_carry <= carry;
49:
50:
51:         process(i_clk)
52:         begin
53:                 --if ( rising_edge(i_clk) and i_enable = '1' ) then
54:                 if ( rising_edge(i_clk) and (reset = '1' or i_enable = '1') ) then
55:                         reset <= '0';
56:                         if ( lowest /= i_lowest ) then
57:                                 lowest <= i_lowest;
58:                                 reset <= '1';
59:                         end if;
60:                         if ( highest /= i_highest ) then
61:                                 highest <= i_highest;
62:                                 reset <= '1';
63:                         end if;
64:                         if ( increment /= i_increment ) then
65:                                 increment <= i_increment;
66:                                 reset <= '1';
67:                         end if;
68:                 end if;
69:         end process;
70:
71:
72:         process(i_clk, i_clear, i_preset)
73:         begin
74:                 if ( i_clear = '1' ) then
75:                         count <= lowest;
76:                         carry <= '0';
77:                 elsif ( i_preset = '1' ) then
78:                         count <= highest;
79:                         --carry <= '1';         -- omitted for now
80:                 elsif ( rising_edge(i_clk) and i_enable = '1' ) then
81:                         if ( reset = '1' ) then
82:                                 count <= lowest;
83:                                 carry <= '0';
84:                         elsif ( count >= highest ) then
85:                                 count <= lowest;
86:                                 carry <= not carry;
87:                         else
88:                                 count <= count + increment;
89:                         end if;
90:                 end if;
91:         end process;
92:
93: end Behavioral;
94:
95:
```

```
    1:
    2: library ieee;
    3: use ieee.std_logic_1164.all;
    4: use ieee.std_logic_arith.all;
    5: use ieee.std_logic_unsigned.all;
    6:
    7:
    8:
    9:
   10: library mcu;
   11: use mcu.pkg_mcu.all;
   12:
   13:
   14: --library fifo;
   15: --use fifo.pkg_fifo_parameters.all;
   16: --use fifo.pkg_fifo.all;
   17:
   18:
   19: library fast_uart;
   20: use fast_uart.pkg_fast_uart.all;
   21:
   22:
   23: entity cpt_fast_uart_rx is
   24:
   25:         generic (
   26:                 DEVICE_ID : std_logic_vector(31 downto 0);
   27:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0)
   28:         );
   29:
   30:         port (
   31:
   32:                  clk : in std_logic;
   33:                  reset : in std_logic;
   34:                  enable : in std_logic;
   35:
   36:                  baud_div : in integer;
   37:
   38:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
   39:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso;
   40:
   41:                  empty : out std_logic;
   42:                  full : out std_logic;
   43:
   44:                  rxd : in std_logic
   45:         );
   46:
   47: end cpt_fast_uart_rx;
   48:
   49: architecture cpt of cpt_fast_uart_rx is
   50:
   51:
   52:         component cpt_uart_rx_fifo is
   53:                 port (
   54:                         CLK                     : in  std_logic;
   55:                         WR_ACK                  : out std_logic;
   56:                         VALID                                           : out std_logic;
   57:                         RST                     : in  std_logic;
   58:                         WR_EN                                   : in  std_logic;
   59:                         RD_EN                   : in  std_logic;
   60:                         DIN                     : in  std_logic_vector(8-1 downto 0);
   61:                         DOUT                    : out std_logic_vector(8-1 downto 0);
   62:                         FULL                    : out std_logic;
   63:                         EMPTY                   : out std_logic
   64:                 );
   65:         end component;
   66:
   67:
   68:         signal fifo_wen : std_logic := '0';
   69:         signal fifo_ren : std_logic := '0';
   70:         signal fifo_in : std_logic_vector(7 downto 0) := (others => '0');
   71:         signal fifo_out : std_logic_vector(7 downto 0);
   72:         signal fifo_word_in : std_logic_vector(15 downto 0) := (others => '0');
   73:         signal fifo_word_out : std_logic_vector(15 downto 0);
   74:         signal fifo_ack : std_logic;
   75:         signal fifo_ack_del : std_logic;
   76:         signal fifo_ack_del2 : std_logic;
   77:         signal fifo_ack_del3 : std_logic;
   78:         signal fifo_valid : std_logic;
   79:         signal fifo_full : std_logic;
   80:         signal fifo_empty : std_logic;
   81:
   82:
   83:
   84:
   85:
   86:         constant RX_LINE_IDLE : std_logic := '1';
   87:         constant START_BIT : std_logic := not RX_LINE_IDLE;
   88:         constant STOP_BIT : std_logic := RX_LINE_IDLE;
   89:
   90:         signal ref : std_logic;
   91:         signal rxd_lpf : std_logic;
   92:         signal last_rxd : std_logic;
   93:
   94:         constant STAGE_RESET : integer := 0;
   95:         constant STAGE_IDLE : integer := 1;
   96:         constant STAGE_HOLDOFF : integer := 2;
   97:         constant STAGE_RX : integer := 10;
   98:
   99:         signal state : integer range 0 to 15;
  100:         signal count : integer;
```

```
101:
102:         signal bitpoint : std_logic_vector(10 downto 0) := "00000000001";
103:
104:         signal dbnc_count : integer;
105:
106:         signal rx_in : std_logic;
107:         signal rx_buf : std_logic_vector(9 downto 0);
108:         signal rx_byte : std_logic_vector(7 downto 0);
109:         signal rx_en : std_logic;
110:         signal rx_we : std_logic;
111:
112:         signal clk_en : std_logic;
113:
114:         signal is_start_bit : std_logic;
115:         signal is_stop_bit : std_logic;
116:
117:
118:         signal bcount : integer;
119:
120: begin
121:
122:
123:
124:         rx_fifo  : cpt_uart_rx_fifo
125:     port map (
126:            CLK                      => clk,
127:            RST                      => reset,
128:            WR_EN                                                => fifo_wen,
129:            RD_EN                    => fifo_ren,
130:            DIN                      => fifo_in,
131:            DOUT                     => fifo_out,
132:            WR_ACK                   => fifo_ack,
133:            VALID                   => fifo_valid,
134:            FULL                     => fifo_full,
135:            EMPTY                    => fifo_empty
136:         );
137:
138:
139:
140:         fifo_ren <= i_mcu_iobus_mosi.read_strobe when ( (i_mcu_iobus_mosi.address and DEVICE_ID_MASK) = (DEVICE_ID and DEVICE_ID_MASK) ) else '0';
141:
142:         fifo_in <= rx_byte;
143:
144:         o_mcu_iobus_miso.read_data <= x"000000" & fifo_out;
145:
146:         fifo_wen <= rx_we;
147:
148:         empty <= fifo_empty;
149:
150:         o_mcu_iobus_miso.ready <= fifo_valid;
151:
152:
153:         rx_byte <= rx_buf(rx_buf'left-1 downto 1);
154:
155:         rx_in <= rxd;
156:
157:         is_start_bit <= '1' when bitpoint(9) = '1' else '0';
158:         is_stop_bit <= '1' when bitpoint(0) = '1' else '0';
159:
160:
161:         -- Baud rate clk_en generator
162:         process (clk)
163:                 begin
164:                         if ( rising_edge(clk) ) then
165:                                 if ( rx_en /= '1' ) then
166:                                         bcount <= 0;
167:                                         clk_en <= '0';
168:                                 elsif (bcount = 0)then
169:                                         bcount <= baud_div;
170:                                         clk_en <= '1';
171:                                 else
172:                                         bcount <= bcount - 1;
173:                                         clk_en <= '0';
174:                                 end if;
175:                         end if;
176:         end process;
177:
178:
179:
180:
181:         process(clk)
182:         begin
183:                 if ( falling_edge(clk) ) then
184:                         if ( reset /= '0' ) then
185:                                 dbnc_count <= 0;
186:                         elsif ( rx_in = last_rxd ) then
187:                                 if ( dbnc_count = 0 ) then
188:                                         rxd_lpf <= rx_in;
189:                                 else
190:                                         dbnc_count <= dbnc_count - 1;
191:                                 end if;
192:                         else
193:                                 last_rxd <= rx_in;
194:                                 dbnc_count <= 0;
195:                         end if;
196:                 end if;
197:         end process;
198:
199:
200:         process(clk)
```

```
201:         begin
202:
203:
204:                 if ( falling_edge(clk) ) then
205:                         if ( reset /= '0' ) then
206:                                 rx_buf <= (others =>RX_LINE_IDLE);
207:                                 rx_en <= '0';
208:                                 rx_we <= '0';
209:                                 state <= STAGE_RESET;
210:                         else
211:                                 case state is
212:
213:                                         when STAGE_RESET =>
214:                                                 rx_buf <= (others => RX_LINE_IDLE);
215:                                                 rx_en <= '0';
216:                                                 rx_we <= '0';
217:                                                 state <= STAGE_IDLE;
218:
219:                                         when STAGE_IDLE =>
220:                                                 rx_buf <= (others => RX_LINE_IDLE);
221:                                                 rx_en <= '0';
222:                                                 rx_we <= '0';
223:                                                 bitpoint <= "00000000001";
224:                                                 if ( rxd_lpf = START_BIT ) then                 -- Wait for a start bit
225:                                                         rx_we <= '0';
226:                                                         bitpoint <= "10000000000";
227:                                                         state <= STAGE_HOLDOFF;
228:                                                         count <= 0;
229:                                                 end if;
230:
231:                                         when STAGE_HOLDOFF =>
232:                                                 if ( count = 0 and rxd_lpf = START_BIT ) then
233:                                                         rx_en <= '1';
234:                                                         state <= STAGE_RX;
235:                                                 elsif ( count = 0 and rxd_lpf /= START_BIT ) then
236:                                                         state <= STAGE_IDLE;
237:                                                 else
238:                                                         count <= count - 1;
239:                                                 end if;
240:
241:                                         -- Shift bits in
242:                                         when STAGE_RX =>
243:                                                 if ( bitpoint(0) = '1' ) then
244:                                                         state <= STAGE_IDLE;
245:                                                         rx_en <= '0';
246:                                                         -- Write rx byte only if a stop bit was seen
247:                                                         if ( rx_buf(rx_buf'left) = STOP_BIT ) then
248:                                                                 rx_we <= '1';
249:                                                         end if;
250:                                                 elsif ( clk_en = '1' ) then
251:                                                         rx_buf <= rxd_lpf & rx_buf(rx_buf'left downto 1);
252:                                                         bitpoint <= bitpoint(0) & bitpoint(10 downto 1);
253:                                                 end if;
254:
255:                                         when others =>
256:                                                 state <= STAGE_IDLE;
257:
258:                                 end case;
259:                         end if;
260:                 end if;
261:         end process;
262:
263:
264: end cpt;
```

```
    1:
    2: library ieee;
    3: use ieee.std_logic_1164.all;
    4: use ieee.std_logic_arith.all;
    5: use ieee.std_logic_unsigned.all;
    6: use ieee.numeric_std.all;
    7: use ieee.math_real.all;
    8:
    9:
   10:
   11: library mcu;
   12: use mcu.pkg_mcu.all;
   13:
   14:
   15: entity cpt_fast_uart_tx is
   16:
   17:         generic (
   18:                 DEVICE_ID : std_logic_vector(31 downto 0);
   19:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0)
   20:         );
   21:
   22:         port (
   23:
   24:                 clk : in std_logic;
   25:                 reset : in std_logic;
   26:                 enable : in std_logic;
   27:
   28:                 baud_div : in integer;
   29:
   30:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
   31:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso;
   32:
   33:                 empty : out std_logic;
   34:                 full : out std_logic;
   35:
   36:                 txd : out std_logic
   37:         );
   38:
   39: end cpt_fast_uart_tx;
   40:
   41:
   42: architecture Behavioral of cpt_fast_uart_tx is
   43:
   44:
   45:         COMPONENT cpt_uart_tx_fifo
   46:           PORT (
   47:                  clk : IN STD_LOGIC;
   48:                  rst : IN STD_LOGIC;
   49:                  din : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
   50:                  wr_en : IN STD_LOGIC;
   51:                  rd_en : IN STD_LOGIC;
   52:                  dout : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   53:                  full : OUT STD_LOGIC;
   54:                  wr_ack : OUT STD_LOGIC;
   55:                  empty : OUT STD_LOGIC;
   56:                  valid : OUT STD_LOGIC
   57:           );
   58:         END COMPONENT;
   59:
   60:
   61:         signal clk_en : std_logic;
   62:
   63:         signal newbyte : std_logic;
   64:         signal newbyte_del : std_logic;
   65:
   66:         signal outbuf : std_logic_vector(9 downto 0) := "1111111111";
   67:
   68:         signal outval : std_logic_vector(7 downto 0);
   69:
   70:         signal bitpoint : std_logic_vector(7 downto 0) := "00000000";
   71:
   72:         --signal count : integer range 0 to 2**16-1;
   73:         signal count : integer;
   74:
   75:         signal fcount : integer range 0 to 15 := 0;
   76:         signal fcount_max : integer range 0 to 15 := 0;
   77:
   78:         signal fifo_wen : std_logic := '0';
   79:         signal fifo_ren : std_logic := '0';
   80:         signal fifo_in : std_logic_vector(7 downto 0) := (others => '0');
   81:         signal fifo_out : std_logic_vector(7 downto 0);
   82:         signal fifo_ack : std_logic;
   83:         signal fifo_full : std_logic;
   84:         signal fifo_empty : std_logic;
   85:
   86:         signal fifo_valid : std_logic;
   87:
   88:
   89:         signal write_data : std_logic_vector(31 downto 0);
   90:         signal write_strobe : std_logic;
   91:
   92:
   93:         signal read_state : std_logic_vector(3 downto 0) := (others => '0');
   94:
   95: begin
   96:
   97:
   98:          tx_fifo  : cpt_uart_tx_fifo
   99:     port map (
  100:            CLK                      => clk,
```

```
101:            RST                     => reset,
102:            WR_EN                                           => fifo_wen,
103:            RD_EN                   => fifo_ren,
104:            DIN                     => fifo_in,
105:            DOUT                    => fifo_out,
106:            WR_ACK                  => fifo_ack,
107:            VALID                   => fifo_valid,
108:            FULL                    => fifo_full,
109:            EMPTY                   => fifo_empty
110:          );
111:
112:
113:
114:
115:          write_data <=   i_mcu_iobus_mosi.write_data;
116:
117:          write_strobe <= i_mcu_iobus_mosi.write_strobe;
118:
119:
120:          fifo_in <= write_data(7 downto 0);
121:
122:          fifo_wen <= write_strobe when ( (i_mcu_iobus_mosi.address and DEVICE_ID_MASK) = (DEVICE_ID and DEVICE_ID_MASK) ) else '0';
123:
124:          fifo_ren <= read_state(2);
125:
126:          o_mcu_iobus_miso.ready <= fifo_ack;
127:
128:          full <= fifo_full;
129:
130:          empty <= fifo_empty;
131:
132:
133:
134:          -- Baud rate clk_en generator
135:          process (clk)
136:                  begin
137:                          if ( enable /= '1' ) then
138:                                          count <= 0;
139:                                          clk_en <= '0';
140:                          elsif ( rising_edge(clk) ) then
141:
142:                             if (count = 0)then
143:                                          count <= baud_div;
144:                                          clk_en <= '1';
145:                                  else
146:                                          count <= count - 1;
147:                                          clk_en <= '0';
148:                                  end if;
149:                          end if;
150:          end process;
151:
152:
153:
154:          process(clk)
155:          begin
156:                  if ( enable /= '1' ) then
157:                                  outbuf <= (others => '0');
158:                                  outval <= (others => '0');
159:                  elsif ( falling_edge(clk)) then
160:
161:                          if ( read_state(2) = '1' ) then
162:                                  outbuf <= '1' & fifo_out & '0';
163:                                  outval <= fifo_out;
164:                          elsif ( clk_en = '1' ) then
165:                                  outbuf <= '1' & outbuf(9 downto 1);
166:                          end if;
167:                  end if;
168:          end process;
169:
170:
171:          process(clk)
172:          begin
173:                  if ( enable /= '1' ) then
174:                                  read_state <= (others => '0');
175:                  elsif ( falling_edge(clk) ) then
176:
177:                          if ( bitpoint = "00000000" and newbyte_del = '0') then
178:                                  read_state <= (clk_en and not fifo_empty) & read_state(3 downto 1);
179:                          else
180:                                  read_state <= '0' & read_state(3 downto 1);
181:                          end if;
182:                  end if;
183:          end process;
184:
185:
186:
187:          process(clk)
188:          begin
189:                  if ( enable /= '1' ) then
190:                                  newbyte <= '0';
191:                  elsif ( falling_edge(clk) ) then
192:
193:                          if ( clk_en = '1' ) then
194:                                  newbyte <= '0';
195:                          elsif ( read_state(3) = '1' ) then
196:                                  newbyte <= '1';
197:                          end if;
198:                  end if;
199:          end process;
200:
```

```
201:         process(clk)
202:         begin
203:                 if ( rising_edge(clk) ) then
204:                         newbyte_del <= newbyte;
205:                 end if;
206:         end process;
207:
208:
209:
210:
211:         process(clk)
212:         begin
213:                 if ( enable /= '1' ) then
214:                                 bitpoint <= (others => '0');
215:                 elsif ( falling_edge(clk)) then
216:
217:                         if (clk_en /= '1') then
218:
219:                         else
220:                                 bitpoint <= newbyte_del & bitpoint(7 downto 1);
221:                         end if;
222:                 end if;
223:         end process;
224:
225:
226:
227:         process(clk)
228:         begin
229:                 if ( enable /= '1' ) then
230:                                 txd <= '1';
231:                 elsif ( falling_edge(clk)) then
232:
233:                         if (clk_en /= '1') then
234:
235:                         else
236:                                 txd <= outbuf(0);
237:                         end if;
238:                 end if;
239:         end process;
240:
241:
242: end Behavioral;
```

```
 1:
 2: library ieee;
 3: use ieee.std_logic_1164.all;
 4:
 5: library mcu;
 6: use mcu.pkg_mcu.all;
 7:
 8:
 9: library fast_uart;
10:
11: package pkg_fast_uart is
12:
13:         component cpt_fast_uart_tx is
14:
15:                 generic (
16:                         DEVICE_ID : std_logic_vector(31 downto 0);
17:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0)
18:                 );
19:
20:
21:                 port (
22:
23:                         clk : in std_logic;
24:                         reset : in std_logic;
25:                         enable : in std_logic;
26:
27:                         baud_div : in integer range 0 to 2**16-1;
28:
29:
30:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
31:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso;
32:
33:                         empty : out std_logic;
34:                         full : out std_logic;
35:
36:                         txd : out std_logic
37:                 );
38:
39:         end component;
40:
41:
42:
43:
44:
45:
46:
47:         component cpt_fast_uart_rx is
48:
49:                 generic (
50:                         DEVICE_ID : std_logic_vector(31 downto 0);
51:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0)
52:                 );
53:
54:                 port (
55:
56:                          clk : in std_logic;
57:                          reset : in std_logic;
58:                          enable : in std_logic;
59:
60:                          baud_div : in integer range 0 to  2**16-1;
61:
62:
63:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
64:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso;
65:
66:                         full : out std_logic;
67:                         empty : out std_logic;
68:
69:                          rxd : in std_logic
70:                 );
71:
72:         end component;
73:
74:
75:
76:
77: end pkg_fast_uart;
```

```
 1:
 2: -- LED multiplexer/driver
 3: --
 4: -- There are 7 user LEDs on-board.
 5: -- These LEDs are driven one at a time using three LED address lines from the FPGA through a 3-to-8 line demultiplexer IC.
 6: -- LED N (1<=N<=7) is enabled when the 3-bit LED address from the FPGA is equal to N.
 7: -- When the LED address is 0, no LED is illuminated.
 8: -- The LED switching frequency is equal to the frequency of i_clk divided by i_led_clk_div.
 9:
10:
11: library ieee;
12: use ieee.std_logic_1164.all;
13: use ieee.std_logic_arith.all;
14:
15: library util;
16: use util.pkg_util.all;
17:
18: library leds;
19: use leds.pkg_leds.all;
20:
21:
22: entity cpt_leds is
23:
24:         port (
25:                 i_clk : in std_logic;
26:                 i_leds : in std_logic_vector(7 downto 1);
27:                 i_led_clk_div : in integer;
28:                 i_led_latch_div : in integer;
29:                 o_led_addr : out std_logic_vector(2 downto 0)
30:         );
31:
32: end cpt_leds;
33:
34:
35: architecture Behavioral of cpt_leds is
36:
37:         signal leds : std_logic_vector(7 downto 1);
38:         signal led_count : integer range 0 to 7 := 1;
39:         signal clk_pgate : std_logic := '0';
40:         signal latch_pgate : std_logic := '0';
41:
42: begin
43:
44:
45:         led_clk_gate : cpt_clk_gate
46:         port map (
47:                 i_clk => i_clk,
48:                 i_enable => '1',
49:                 i_div => i_led_clk_div,
50:                 o_clk_pgate => clk_pgate,
51:                 o_clk_ngate => open
52:         );
53:
54:         led_latch_gate : cpt_clk_gate
55:         port map (
56:                 i_clk => i_clk,
57:                 i_enable => '1',
58:                 i_div => i_led_latch_div,
59:                 o_clk_pgate => latch_pgate,
60:                 o_clk_ngate => open
61:         );
62:
63:         process(i_clk)
64:         begin
65:                 if ( rising_edge(i_clk) and latch_pgate = '1' ) then
66:                         leds <= i_leds;
67:                 end if;
68:         end process;
69:
70:
71:         o_led_addr <= conv_std_logic_vector(led_count, 3) when leds(led_count) = '1' else "000";
72:
73:         led_counter : cpt_upcounter
74:         generic map (
75:                 INIT => 1
76:         )
77:         port map (
78:                 i_clk => i_clk,
79:                 i_enable => clk_pgate,
80:                 i_clear => '0',
81:                 i_preset => '0',
82:                 i_increment => 1,
83:                 i_lowest => 1,
84:                 i_highest => 7,
85:                 o_count => led_count,
86:                 o_carry => open
87:         );
88:
89:
90: end Behavioral;
91:
```

```
 1:
 2: library ieee;
 3: use ieee.std_logic_1164.all;
 4:
 5: library leds;
 6:
 7: package pkg_leds is
 8:
 9:         component cpt_leds is
10:                 port (
11:                         i_clk : in std_logic;
12:                         i_leds : in std_logic_vector(7 downto 1);
13:                         i_led_clk_div : in integer;
14:                         i_led_latch_div : in integer;
15:                         o_led_addr : out std_logic_vector(2 downto 0)
16:                 );
17:         end component;
18:
19:
20: end pkg_leds;
21:
22:
23: package body pkg_leds is
24: end pkg_leds;
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5: library mctl;
  6: use mctl.pkg_mctl.all;
  7:
  8: entity cpt_mctl_wrapper is
  9:
 10:         generic (
 11:                 INCLUDE_MCTL_CHIPSCOPE : string := "TRUE";
 12:                 C3_MEMCLK_PERIOD        : integer := 6944; -- 144 MHz
 13:                 C3_CALIB_SOFT_IP        : string := "TRUE";
 14:                 C3_SIMULATION           : string := "FALSE"
 15:         );
 16:
 17:         port (
 18:
 19:                 mcb3_rzq : inout  std_logic;
 20:
 21:                 c3_sys_clk : in  std_logic;
 22:                 c3_sys_rst_i : in  std_logic;
 23:                 c3_calib_done : out std_logic;
 24:                 c3_clk0 : out std_logic;
 25:                 c3_rst0 : out std_logic;
 26:
 27:
 28:                         clk_108         : out std_logic;
 29:                         clk_108_n       : out std_logic;
 30:
 31:                 ram_bidir : inout typ_mctl_ram_bidir;
 32:                 ram_mosi : out typ_mctl_ram_mosi;
 33:
 34:                 mport0_miso : in typ_mctl_mport_miso;
 35:                 mport0_mosi : out typ_mctl_mport_mosi;
 36:
 37:                 mport1_miso : in typ_mctl_mport_miso;
 38:                 mport1_mosi : out typ_mctl_mport_mosi;
 39:
 40:                 mport2_miso : in typ_mctl_mport_miso;
 41:                 mport2_mosi : out typ_mctl_mport_mosi;
 42:
 43:                 mport3_miso : in typ_mctl_mport_miso;
 44:                 mport3_mosi : out typ_mctl_mport_mosi
 45:         );
 46:
 47: end cpt_mctl_wrapper;
 48:
 49:
 50: architecture arc of cpt_mctl_wrapper is
 51:
 52: begin
 53:
 54:
 55:
 56:         mctl : cpt_mctl
 57:
 58:         generic map (
 59:                 INCLUDE_MCTL_CHIPSCOPE => INCLUDE_MCTL_CHIPSCOPE,
 60:                 C3_P0_MASK_SIZE => 4,
 61:                 C3_P0_DATA_PORT_SIZE => 32,
 62:                 C3_P1_MASK_SIZE => 4,
 63:                 C3_P1_DATA_PORT_SIZE => 32,
 64:                 C3_MEMCLK_PERIOD => C3_MEMCLK_PERIOD,
 65:                 C3_RST_ACT_LOW => 0,
 66:                 C3_INPUT_CLK_TYPE => "SINGLE_ENDED",
 67:                 C3_CALIB_SOFT_IP => C3_CALIB_SOFT_IP,
 68:                 C3_SIMULATION => C3_SIMULATION,
 69:                 DEBUG_EN => 0,
 70:                 C3_MEM_ADDR_ORDER => "ROW_BANK_COLUMN",
 71:                 C3_NUM_DQ_PINS => 16,
 72:                 C3_MEM_ADDR_WIDTH => 13,
 73:                 C3_MEM_BANKADDR_WIDTH => 2
 74:         )
 75:
 76:         port map (
 77:
 78:
 79:                 mcb3_dram_dq       =>    ram_bidir.dq,
 80:                 mcb3_dram_dqs      =>    ram_bidir.dqs,
 81:                 mcb3_dram_udqs  =>       ram_bidir.udqs,
 82:
 83:                 mcb3_dram_a        =>    ram_mosi.a,
 84:                 mcb3_dram_ba       =>    ram_mosi.ba,
 85:                 mcb3_dram_ras_n    =>    ram_mosi.ras_n,
 86:                 mcb3_dram_cas_n    =>    ram_mosi.cas_n,
 87:                 mcb3_dram_we_n     =>    ram_mosi.we_n,
 88:                 mcb3_dram_cke      =>    ram_mosi.cke,
 89:                 mcb3_dram_ck       =>    ram_mosi.ck,
 90:                 mcb3_dram_ck_n     =>    ram_mosi.ck_n,
 91:                 mcb3_dram_udm  =>        ram_mosi.udm,
 92:                 mcb3_dram_dm  =>       ram_mosi.dm,
 93:
 94:
 95:                 c3_sys_clk => c3_sys_clk,
 96:                 c3_sys_rst_i => c3_sys_rst_i,
 97:
 98:                 c3_clk0 => c3_clk0,
 99:                 c3_rst0 => c3_rst0,
100:
```

```
101:                c3_calib_done => c3_calib_done,
102:
103:                mcb3_rzq => mcb3_rzq,
104:
105:
106:
107:    clk_108   => clk_108,
108:    clk_108_n => clk_108_n,
109:
110:                c3_p0_cmd_clk                         =>  mport0_miso.cmd.clk,
111:                c3_p0_cmd_en                          =>  mport0_miso.cmd.en,
112:                c3_p0_cmd_instr                       =>  mport0_miso.cmd.instr,
113:                c3_p0_cmd_bl                          =>  mport0_miso.cmd.bl,
114:                c3_p0_cmd_byte_addr                   =>  mport0_miso.cmd.byte_addr,
115:                c3_p0_cmd_empty                       =>  mport0_mosi.cmd.empty,
116:                c3_p0_cmd_full                        =>  mport0_mosi.cmd.full,
117:                c3_p0_wr_clk                          =>  mport0_miso.wr.clk,
118:                c3_p0_wr_en                           =>  mport0_miso.wr.en,
119:                c3_p0_wr_mask                         =>  mport0_miso.wr.mask,
120:                c3_p0_wr_data                         =>  mport0_miso.wr.data,
121:                c3_p0_wr_full                         =>  mport0_mosi.wr.full,
122:                c3_p0_wr_empty                        =>  mport0_mosi.wr.empty,
123:                c3_p0_wr_count                        =>  mport0_mosi.wr.count,
124:                c3_p0_wr_underrun                     =>  mport0_mosi.wr.underrun,
125:                c3_p0_wr_error                        =>  mport0_mosi.wr.error,
126:                c3_p0_rd_clk                          =>  mport0_miso.rd.clk,
127:                c3_p0_rd_en                           =>  mport0_miso.rd.en,
128:                c3_p0_rd_data                         =>  mport0_mosi.rd.data,
129:                c3_p0_rd_full                         =>  mport0_mosi.rd.full,
130:                c3_p0_rd_empty                        =>  mport0_mosi.rd.empty,
131:                c3_p0_rd_count                        =>  mport0_mosi.rd.count,
132:                c3_p0_rd_overflow                     =>  mport0_mosi.rd.overflow,
133:                c3_p0_rd_error                        =>  mport0_mosi.rd.error,
134:                c3_p1_cmd_clk                         =>  mport1_miso.cmd.clk,
135:                c3_p1_cmd_en                          =>  mport1_miso.cmd.en,
136:                c3_p1_cmd_instr                       =>  mport1_miso.cmd.instr,
137:                c3_p1_cmd_bl                          =>  mport1_miso.cmd.bl,
138:                c3_p1_cmd_byte_addr                   =>  mport1_miso.cmd.byte_addr,
139:                c3_p1_cmd_empty                       =>  mport1_mosi.cmd.empty,
140:                c3_p1_cmd_full                        =>  mport1_mosi.cmd.full,
141:                c3_p1_wr_clk                          =>  mport1_miso.wr.clk,
142:                c3_p1_wr_en                           =>  mport1_miso.wr.en,
143:                c3_p1_wr_mask                         =>  mport1_miso.wr.mask,
144:                c3_p1_wr_data                         =>  mport1_miso.wr.data,
145:                c3_p1_wr_full                         =>  mport1_mosi.wr.full,
146:                c3_p1_wr_empty                        =>  mport1_mosi.wr.empty,
147:                c3_p1_wr_count                        =>  mport1_mosi.wr.count,
148:                c3_p1_wr_underrun                     =>  mport1_mosi.wr.underrun,
149:                c3_p1_wr_error                        =>  mport1_mosi.wr.error,
150:                c3_p1_rd_clk                          =>  mport1_miso.rd.clk,
151:                c3_p1_rd_en                           =>  mport1_miso.rd.en,
152:                c3_p1_rd_data                         =>  mport1_mosi.rd.data,
153:                c3_p1_rd_full                         =>  mport1_mosi.rd.full,
154:                c3_p1_rd_empty                        =>  mport1_mosi.rd.empty,
155:                c3_p1_rd_count                        =>  mport1_mosi.rd.count,
156:                c3_p1_rd_overflow                     =>  mport1_mosi.rd.overflow,
157:                c3_p1_rd_error                        =>  mport1_mosi.rd.error,
158:                c3_p2_cmd_clk                         =>  mport2_miso.cmd.clk,
159:                c3_p2_cmd_en                          =>  mport2_miso.cmd.en,
160:                c3_p2_cmd_instr                       =>  mport2_miso.cmd.instr,
161:                c3_p2_cmd_bl                          =>  mport2_miso.cmd.bl,
162:                c3_p2_cmd_byte_addr                   =>  mport2_miso.cmd.byte_addr,
163:                c3_p2_cmd_empty                       =>  mport2_mosi.cmd.empty,
164:                c3_p2_cmd_full                        =>  mport2_mosi.cmd.full,
165:                c3_p2_wr_clk                          =>  mport2_miso.wr.clk,
166:                c3_p2_wr_en                           =>  mport2_miso.wr.en,
167:                c3_p2_wr_mask                         =>  mport2_miso.wr.mask,
168:                c3_p2_wr_data                         =>  mport2_miso.wr.data,
169:                c3_p2_wr_full                         =>  mport2_mosi.wr.full,
170:                c3_p2_wr_empty                        =>  mport2_mosi.wr.empty,
171:                c3_p2_wr_count                        =>  mport2_mosi.wr.count,
172:                c3_p2_wr_underrun                     =>  mport2_mosi.wr.underrun,
173:                c3_p2_wr_error                        =>  mport2_mosi.wr.error,
174:                c3_p2_rd_clk                          =>  mport2_miso.rd.clk,
175:                c3_p2_rd_en                           =>  mport2_miso.rd.en,
176:                c3_p2_rd_data                         =>  mport2_mosi.rd.data,
177:                c3_p2_rd_full                         =>  mport2_mosi.rd.full,
178:                c3_p2_rd_empty                        =>  mport2_mosi.rd.empty,
179:                c3_p2_rd_count                        =>  mport2_mosi.rd.count,
180:                c3_p2_rd_overflow                     =>  mport2_mosi.rd.overflow,
181:                c3_p2_rd_error                        =>  mport2_mosi.rd.error,
182:                c3_p3_cmd_clk                         =>  mport3_miso.cmd.clk,
183:                c3_p3_cmd_en                          =>  mport3_miso.cmd.en,
184:                c3_p3_cmd_instr                       =>  mport3_miso.cmd.instr,
185:                c3_p3_cmd_bl                          =>  mport3_miso.cmd.bl,
186:                c3_p3_cmd_byte_addr                   =>  mport3_miso.cmd.byte_addr,
187:                c3_p3_cmd_empty                       =>  mport3_mosi.cmd.empty,
188:                c3_p3_cmd_full                        =>  mport3_mosi.cmd.full,
189:                c3_p3_wr_clk                          =>  mport3_miso.wr.clk,
190:                c3_p3_wr_en                           =>  mport3_miso.wr.en,
191:                c3_p3_wr_mask                         =>  mport3_miso.wr.mask,
192:                c3_p3_wr_data                         =>  mport3_miso.wr.data,
193:                c3_p3_wr_full                         =>  mport3_mosi.wr.full,
194:                c3_p3_wr_empty                        =>  mport3_mosi.wr.empty,
195:                c3_p3_wr_count                        =>  mport3_mosi.wr.count,
196:                c3_p3_wr_underrun                     =>  mport3_mosi.wr.underrun,
197:                c3_p3_wr_error                        =>  mport3_mosi.wr.error,
198:                c3_p3_rd_clk                          =>  mport3_miso.rd.clk,
199:                c3_p3_rd_en                           =>  mport3_miso.rd.en,
200:                c3_p3_rd_data                         =>  mport3_mosi.rd.data,
```

```
201:                c3_p3_rd_full                       =>  mport3_mosi.rd.full,
202:                c3_p3_rd_empty                      =>  mport3_mosi.rd.empty,
203:                c3_p3_rd_count                      =>  mport3_mosi.rd.count,
204:                c3_p3_rd_overflow                   =>  mport3_mosi.rd.overflow,
205:                c3_p3_rd_error                      =>  mport3_mosi.rd.error
206:        );
207:
208:  end  arc;
```

```
   1: --*****************************************************************************
   2: -- (c) Copyright 2009 Xilinx, Inc. All rights reserved.
   3: --
   4: -- This file contains confidential and proprietary information
   5: -- of Xilinx, Inc. and is protected under U.S. and
   6: -- international copyright and other intellectual property
   7: -- laws.
   8: --
   9: -- DISCLAIMER
  10: -- This disclaimer is not a license and does not grant any
  11: -- rights to the materials distributed herewith. Except as
  12: -- otherwise provided in a valid license issued to you by
  13: -- Xilinx, and to the maximum extent permitted by applicable
  14: -- law: (1) THESE MATERIALS ARE MADE AVAILABLE "AS IS" AND
  15: -- WITH ALL FAULTS, AND XILINX HEREBY DISCLAIMS ALL WARRANTIES
  16: -- AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING
  17: -- BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-
  18: -- INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and
  19: -- (2) Xilinx shall not be liable (whether in contract or tort,
  20: -- including negligence, or under any other theory of
  21: -- liability) for any loss or damage of any kind or nature
  22: -- related to, arising under or in connection with these
  23: -- materials, including for any direct, or any indirect,
  24: -- special, incidental, or consequential loss or damage
  25: -- (including loss of data, profits, goodwill, or any type of
  26: -- loss or damage suffered as a result of any action brought
  27: -- by a third party) even if such damage or loss was
  28: -- reasonably foreseeable or Xilinx had been advised of the
  29: -- possibility of the same.
  30: --
  31: -- CRITICAL APPLICATIONS
  32: -- Xilinx products are not designed or intended to be fail-
  33: -- safe, or for use in any application requiring fail-safe
  34: -- performance, such as life-support or safety devices or
  35: -- systems, Class III medical devices, nuclear facilities,
  36: -- applications related to the deployment of airbags, or any
  37: -- other applications that could lead to death, personal
  38: -- injury, or severe property or environmental damage
  39: -- (individually and collectively, "Critical
  40: -- Applications"). Customer assumes the sole risk and
  41: -- liability of any use of Xilinx products in Critical
  42: -- Applications, subject only to applicable laws and
  43: -- regulations governing limitations on product liability.
  44: --
  45: -- THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS
  46: -- PART OF THIS FILE AT ALL TIMES.
  47: --
  48: --*****************************************************************************
  49: --   ____  ____
  50: --  /   /\/   /
  51: -- /___/  \  /    Vendor             : Xilinx
  52: -- \   \   \/     Version            : 3.92
  53: --  \   \         Application        : MIG
  54: --  /   /         Filename           : memc3_infrastructure.vhd
  55: -- /___/   /\     Date Last Modified : $Date: 2011/06/02 07:17:24 $
  56: -- \   \  /  \    Date Created       : Jul 03 2009
  57: --  \___\/\___\
  58: --
  59: --Device           : Spartan-6
  60: --Design Name      : DDR/DDR2/DDR3/LPDDR
  61: --Purpose          : Clock generation/distribution and reset synchronization
  62: --Reference        :
  63: --Revision History :
  64: --*****************************************************************************
  65: library ieee;
  66: use ieee.std_logic_1164.all;
  67: library unisim;
  68: use unisim.vcomponents.all;
  69:
  70: entity memc3_infrastructure is
  71: generic
  72:   (
  73:     C_INCLK_PERIOD     : integer := 2500;
  74:     C_RST_ACT_LOW      : integer := 1;
  75:     C_INPUT_CLK_TYPE   : string  := "DIFFERENTIAL";
  76:     C_CLKOUT0_DIVIDE   : integer := 2;
  77:     C_CLKOUT1_DIVIDE   : integer := 2;
  78:     C_CLKOUT2_DIVIDE   : integer := 8;
  79:     C_CLKOUT3_DIVIDE   : integer := 8;
  80:     C_CLKOUT4_DIVIDE   : integer := 4;
  81:     C_CLKOUT5_DIVIDE   : integer := 4;
  82:     C_CLKFBOUT_MULT   : integer := 4;
  83:     C_DIVCLK_DIVIDE   : integer := 1
  84:
  85:   );
  86: port
  87: (
  88:     sys_clk_p       : in std_logic;
  89:     sys_clk_n       : in std_logic;
  90:     sys_clk       : in std_logic;
  91:     sys_rst_i       : in std_logic;
  92:     clk0            : out std_logic;
  93:     rst0            : out std_logic;
  94:     async_rst       : out std_logic;
  95:     sysclk_2x       : out std_logic;
  96:     sysclk_2x_180   : out std_logic;
  97:     mcb_drp_clk     : out std_logic;
  98:     clk_108         : out std_logic;
  99:     clk_108_n       : out std_logic;
 100:     pll_ce_0        : out std_logic;
```

```
101:     pll_ce_90       : out std_logic;
102:     pll_lock        : out std_logic
103:
104: );
105: end entity;
106: architecture syn of memc3_infrastructure is
107:
108:   -- # of clock cycles to delay deassertion of reset. Needs to be a fairly
109:   -- high number not so much for metastability protection, but to give time
110:   -- for reset (i.e. stable clock cycles) to propagate through all state
111:   -- machines and to all control signals (i.e. not all control signals have
112:   -- resets, instead they rely on base state logic being reset, and the effect
113:   -- of that reset propagating through the logic). Need this because we may not
114:   -- be getting stable clock cycles while reset asserted (i.e. since reset
115:   -- depends on PLL/DCM lock status)
116:
117:   constant RST_SYNC_NUM   : integer := 25;
118:   constant CLK_PERIOD_NS  : real := (real(C_INCLK_PERIOD)) / 1000.0;
119:   constant CLK_PERIOD_INT : integer := C_INCLK_PERIOD/1000;
120:
121:
122:   signal   clk_2x_0            : std_logic;
123:   signal   clk_2x_180          : std_logic;
124:   signal   clk0_bufg           : std_logic;
125:   signal   clk0_bufg_in        : std_logic;
126:   signal   mcb_drp_clk_bufg_in : std_logic;
127:   signal   clkfbout_clkfbin    : std_logic;
128:   signal   rst_tmp             : std_logic;
129:   signal   sys_clk_ibufg       : std_logic;
130:   signal   sys_rst             : std_logic;
131:   signal   rst0_sync_r         : std_logic_vector(RST_SYNC_NUM-1 downto 0);
132:   signal   powerup_pll_locked  : std_logic;
133:   signal   syn_clk0_powerup_pll_locked : std_logic;
134:   signal   locked              : std_logic;
135:   signal   bufpll_mcb_locked   : std_logic;
136:   signal   mcb_drp_clk_sig     : std_logic;
137:
138:   attribute max_fanout : string;
139:   attribute syn_maxfan : integer;
140:   attribute KEEP : string;
141:   attribute max_fanout of rst0_sync_r : signal is "10";
142:   attribute syn_maxfan of rst0_sync_r : signal is 10;
143:   attribute KEEP of sys_clk_ibufg     : signal is "TRUE";
144:
145: begin
146:
147:   sys_rst  <= not(sys_rst_i) when (C_RST_ACT_LOW /= 0) else sys_rst_i;
148:   clk0     <= clk0_bufg;
149:   pll_lock <= bufpll_mcb_locked;
150:   mcb_drp_clk <= mcb_drp_clk_sig;
151:
152:   diff_input_clk : if(C_INPUT_CLK_TYPE = "DIFFERENTIAL") generate
153:       --***********************************************************************
154:       -- Differential input clock input buffers
155:       --***********************************************************************
156:       u_ibufg_sys_clk : IBUFGDS
157:         generic map (
158:           DIFF_TERM => TRUE
159:         )
160:         port map (
161:           I  => sys_clk_p,
162:           IB => sys_clk_n,
163:           O  => sys_clk_ibufg
164:           );
165:   end generate;
166:
167:
168:   se_input_clk : if(C_INPUT_CLK_TYPE = "SINGLE_ENDED") generate
169:       --***********************************************************************
170:       -- SINGLE_ENDED input clock input buffers
171:       --***********************************************************************
172:
173:                -- QuadCam modification: IBUFG moved to top level
174: --      u_ibufg_sys_clk : IBUFG
175: --        port map (
176: --          I  => sys_clk,
177: --          O  => sys_clk_ibufg
178: --          );
179:
180:                         sys_clk_ibufg <= sys_clk;
181:
182:   end generate;
183:
184:   --***********************************************************************
185:   -- Global clock generation and distribution
186:   --***********************************************************************
187:
188:     u_pll_adv : PLL_ADV
189:     generic map
190:         (
191:          BANDWIDTH          => "OPTIMIZED",
192:          CLKIN1_PERIOD      => CLK_PERIOD_NS,
193:          CLKIN2_PERIOD      => CLK_PERIOD_NS,
194:          CLKOUT0_DIVIDE     => C_CLKOUT0_DIVIDE,
195:          CLKOUT1_DIVIDE     => C_CLKOUT1_DIVIDE,
196:          CLKOUT2_DIVIDE     => C_CLKOUT2_DIVIDE,
197:          CLKOUT3_DIVIDE     => C_CLKOUT3_DIVIDE,
198:          CLKOUT4_DIVIDE     => C_CLKOUT4_DIVIDE,
199:          CLKOUT5_DIVIDE     => C_CLKOUT5_DIVIDE,
200:          CLKOUT0_PHASE      => 0.000,
```

```
201:          CLKOUT1_PHASE      => 180.000,
202:          CLKOUT2_PHASE      => 0.000,
203:          CLKOUT3_PHASE      => 0.000,
204:          CLKOUT4_PHASE      => 0.000,
205:          CLKOUT5_PHASE      => 180.000,
206:          CLKOUT0_DUTY_CYCLE => 0.500,
207:          CLKOUT1_DUTY_CYCLE => 0.500,
208:          CLKOUT2_DUTY_CYCLE => 0.500,
209:          CLKOUT3_DUTY_CYCLE => 0.500,
210:          CLKOUT4_DUTY_CYCLE => 0.500,
211:          CLKOUT5_DUTY_CYCLE => 0.500,
212:          SIM_DEVICE         => "SPARTAN6",
213:          COMPENSATION       => "INTERNAL",
214:          DIVCLK_DIVIDE      => C_DIVCLK_DIVIDE,
215:          CLKFBOUT_MULT      => C_CLKFBOUT_MULT,
216:          CLKFBOUT_PHASE     => 0.0,
217:          REF_JITTER         => 0.005000
218:          )
219:         port map
220:           (
221:            CLKFBIN          => clkfbout_clkfbin,
222:            CLKINSEL         => '1',
223:            CLKIN1           => sys_clk_ibufg,
224:            CLKIN2           => '0',
225:            DADDR            => (others => '0'),
226:            DCLK             => '0',
227:            DEN              => '0',
228:            DI               => (others => '0'),
229:            DWE              => '0',
230:            REL              => '0',
231:            RST              => sys_rst,
232:            CLKFBDCM         => open,
233:            CLKFBOUT         => clkfbout_clkfbin,
234:            CLKOUTDCM0       => open,
235:            CLKOUTDCM1       => open,
236:            CLKOUTDCM2       => open,
237:            CLKOUTDCM3       => open,
238:            CLKOUTDCM4       => open,
239:            CLKOUTDCM5       => open,
240:            CLKOUT0          => clk_2x_0,
241:            CLKOUT1          => clk_2x_180,
242:            CLKOUT5          => clk0_bufg_in,
243:            CLKOUT4          => mcb_drp_clk_bufg_in,
244:            CLKOUT2          => clk_108,
245:            CLKOUT3          => clk_108_n,
246:            DO               => open,
247:            DRDY             => open,
248:            LOCKED           => locked
249:            );
250:
251:     U_BUFG_CLK0 : BUFG
252:     port map
253:     (
254:      O => clk0_bufg,
255:      I => clk0_bufg_in
256:      );
257:
258:     --U_BUFG_CLK1 : BUFG
259:     -- port map (
260:     --  O => mcb_drp_clk_sig,
261:     --  I => mcb_drp_clk_bufg_in
262:     --  );
263:
264:     U_BUFG_CLK1 : BUFGCE
265:      port map (
266:       O => mcb_drp_clk_sig,
267:       I => mcb_drp_clk_bufg_in,
268:       CE => locked
269:       );
270:
271:     process (mcb_drp_clk_sig, sys_rst)
272:     begin
273:        if(sys_rst = '1') then
274:           powerup_pll_locked <= '0';
275:        elsif (mcb_drp_clk_sig'event and mcb_drp_clk_sig = '1') then
276:           if (bufpll_mcb_locked = '1') then
277:              powerup_pll_locked <= '1';
278:           end if;
279:        end if;
280:     end process;
281:
282:
283:     process (clk0_bufg, sys_rst)
284:     begin
285:        if(sys_rst = '1') then
286:           syn_clk0_powerup_pll_locked <= '0';
287:        elsif (clk0_bufg'event and clk0_bufg = '1') then
288:           if (bufpll_mcb_locked = '1') then
289:              syn_clk0_powerup_pll_locked <= '1';
290:           end if;
291:        end if;
292:     end process;
293:
294:
295:     --**************************************************************************
296:     -- Reset synchronization
297:     -- NOTES:
298:     --   1. shut down the whole operation if the PLL hasn't yet locked (and
299:     --      by inference, this means that external sys_rst has been asserted -
300:     --      PLL deasserts LOCKED as soon as sys_rst asserted)
```

```
301:    --   2. asynchronously assert reset. This was we can assert reset even if
302:    --      there is no clock (needed for things like 3-stating output buffers).
303:    --      reset deassertion is synchronous.
304:    --   3. asynchronous reset only look at pll_lock from PLL during power up. After
305:    --      power up and pll_lock is asserted, the powerup_pll_locked will be asserted
306:    --      forever until sys_rst is asserted again. PLL will lose lock when FPGA
307:    --      enters suspend mode. We don't want reset to MCB get
308:    --      asserted in the application that needs suspend feature.
309:    --**************************************************************************
310:
311:
312:   async_rst <= sys_rst or not(powerup_pll_locked);
313:   -- async_rst <= rst_tmp;
314:   rst_tmp <= sys_rst or not(syn_clk0_powerup_pll_locked);
315:   -- rst_tmp <= sys_rst or not(powerup_pll_locked);
316:
317: process (clk0_bufg, rst_tmp)
318:   begin
319:     if (rst_tmp = '1') then
320:       rst0_sync_r <= (others => '1');
321:     elsif (rising_edge(clk0_bufg)) then
322:       rst0_sync_r <= rst0_sync_r(RST_SYNC_NUM-2 downto 0) & '0';  -- logical left shift by one (pads with 0)
323:     end if;
324:   end process;
325:
326:   rst0    <= rst0_sync_r(RST_SYNC_NUM-1);
327:
328:
329: BUFPLL_MCB_INST : BUFPLL_MCB
330: port map
331: ( IOCLK0         => sysclk_2x,
332:   IOCLK1         => sysclk_2x_180,
333:   LOCKED         => locked,
334:   GCLK           => mcb_drp_clk_sig,
335:   SERDESSTROBE0  => pll_ce_0,
336:   SERDESSTROBE1  => pll_ce_90,
337:   PLLIN0         => clk_2x_0,
338:   PLLIN1         => clk_2x_180,
339:   LOCK           => bufpll_mcb_locked
340:   );
341:
342: end architecture syn;
343:
```

```
--
-- Memory controller package
-- Reference ug388_Spartan6_MemoryControlBlock.pdf
--

library ieee;
use ieee.std_logic_1164.all;

library mctl;

package pkg_mctl is
	constant C3_NUM_DQ_PINS         : integer := 16;        -- External memory data width.
	constant C3_MEM_ADDR_WIDTH      : integer := 13;        -- External memory address width.
	constant C3_MEM_BANKADDR_WIDTH  : integer := 2;         -- External memory bank address width.
	constant C3_MEM_ADDR_ORDER      : string  := "ROW_BANK_COLUMN";
	constant C3_P0_MASK_SIZE        : integer := 4;
	constant C3_P0_DATA_PORT_SIZE   : integer := 32;
	constant C3_P1_MASK_SIZE        : integer := 4;
	constant C3_P1_DATA_PORT_SIZE   : integer := 32;


	type typ_mctl_mport_cmd_miso is record
		clk : std_logic;
		en : std_logic;
		instr : std_logic_vector(2 downto 0);
		bl : std_logic_vector(5 downto 0);
		byte_addr : std_logic_vector(29 downto 0);
	end record;

	constant init_mctl_mport_cmd_miso : typ_mctl_mport_cmd_miso := (
		clk => '0',
		en => '0',
		instr => (others => '0'),
		bl => (others => '0'),
		byte_addr => (others => '0')
	);


	type typ_mctl_mport_wr_miso is record
		clk : std_logic;
		en : std_logic;
		mask : std_logic_vector(3 downto 0);
		data : std_logic_vector(31 downto 0);
	end record;

	constant init_mctl_mport_wr_miso : typ_mctl_mport_wr_miso := (
		clk => '0',
		en => '0',
		mask => (others => '0'),
		data => (others => '0')
	);


	type typ_mctl_mport_rd_miso is record
		clk : std_logic;
		en : std_logic;
	end record;

	constant init_mctl_mport_rd_miso : typ_mctl_mport_rd_miso := (
		clk => '0',
		en => '0'
	);


	type typ_mctl_mport_miso is record
		cmd : typ_mctl_mport_cmd_miso;
		wr : typ_mctl_mport_wr_miso;
		rd : typ_mctl_mport_rd_miso;
	end record;

	constant init_mctl_mport_miso : typ_mctl_mport_miso := (
		cmd => init_mctl_mport_cmd_miso,
		wr => init_mctl_mport_wr_miso,
		rd => init_mctl_mport_rd_miso
	);


	type typ_mctl_mport_cmd_mosi is record
		empty : std_logic;
		full : std_logic;
	end record;

	constant init_mctl_mport_cmd_mosi : typ_mctl_mport_cmd_mosi := (
		empty => '0',
		full => '0'
	);


	type typ_mctl_mport_wr_mosi is record
		empty : std_logic;
		full : std_logic;
		count : std_logic_vector(6 downto 0);
		underrun : std_logic;
		error : std_logic;
	end record;

	constant init_mctl_mport_wr_mosi : typ_mctl_mport_wr_mosi := (
		empty => '0',
		full => '0',
		count => (others => '0'),
```

```
101:                underrun => '0',
102:                error => '0'
103:        );
104:
105:
106:        type typ_mctl_mport_rd_mosi is record
107:                data : std_logic_vector(31 downto 0);
108:                empty : std_logic;
109:                full : std_logic;
110:                count : std_logic_vector(6 downto 0);
111:                overflow : std_logic;
112:                error : std_logic;
113:        end record;
114:
115:        constant init_mctl_mport_rd_mosi : typ_mctl_mport_rd_mosi := (
116:                data => (others => '0'),
117:                empty => '0',
118:                full => '0',
119:                count => (others => '0'),
120:                overflow => '0',
121:                error => '0'
122:        );
123:
124:
125:        type typ_mctl_mport_mosi is record
126:                cmd : typ_mctl_mport_cmd_mosi;
127:                wr : typ_mctl_mport_wr_mosi;
128:                rd : typ_mctl_mport_rd_mosi;
129:        end record;
130:
131:        constant init_mctl_mport_mosi : typ_mctl_mport_mosi := (
132:                cmd => init_mctl_mport_cmd_mosi,
133:                wr => init_mctl_mport_wr_mosi,
134:                rd => init_mctl_mport_rd_mosi
135:        );
136:
137:
138:        type typ_mctl_ram_bidir is record
139:                dq : std_logic_vector(C3_NUM_DQ_PINS-1 downto 0);
140:                udqs : std_logic;
141:                dqs : std_logic;
142:        end record;
143:
144:        type typ_mctl_ram_mosi is record
145:                a : std_logic_vector(C3_MEM_ADDR_WIDTH-1 downto 0);
146:                ba : std_logic_vector(C3_MEM_BANKADDR_WIDTH-1 downto 0);
147:                cke : std_logic;
148:                ras_n : std_logic;
149:                cas_n : std_logic;
150:                we_n : std_logic;
151:                dm : std_logic;
152:                udm : std_logic;
153:                ck : std_logic;
154:                ck_n : std_logic;
155:        end record;
156:
157:        component cpt_mctl_wrapper is
158:                generic (
159:                        INCLUDE_MCTL_CHIPSCOPE  : string  := "TRUE";
160:                        C3_MEMCLK_PERIOD        : integer := 6000;
161:                                                                                    -- Memory data transfer clock period.
162:                        C3_CALIB_SOFT_IP        : string := "TRUE";
163:                                                                                    -- # = TRUE, Enables the soft calibration logic,
164:                                                                                    -- # = FALSE, Disables the soft calibration logic.
165:                        C3_SIMULATION           : string := "FALSE"
166:                                                                                    -- # = TRUE, Simulating the design. Useful to reduce
the simulation time,
167:                                                                                    -- # = FALSE, Implementing the design.
168:                        --DEBUG_EN                : integer := 1
169:                                                                                    -- # = 1, Enable debug signals/controls,
170:                                                                                    --   = 0, Disable debug signals/controls.
171:                );
172:                port (
173:                        mcb3_rzq                : inout std_logic;
174:
175:                        c3_sys_clk              : in std_logic;
176:                        c3_sys_rst_i            : in std_logic;
177:                        c3_calib_done           : out std_logic;
178:                        c3_clk0                 : out std_logic;
179:                        c3_rst0                 : out std_logic;
180:
181:                        clk_108                 : out std_logic;
182:                        clk_108_n               : out std_logic;
183:
184:                        ram_bidir   : inout typ_mctl_ram_bidir;
185:                        ram_mosi    : out typ_mctl_ram_mosi;
186:
187:                        mport0_miso : in typ_mctl_mport_miso;
188:                        mport0_mosi : out typ_mctl_mport_mosi;
189:
190:                        mport1_miso : in typ_mctl_mport_miso;
191:                        mport1_mosi : out typ_mctl_mport_mosi;
192:
193:                        mport2_miso : in typ_mctl_mport_miso;
194:                        mport2_mosi : out typ_mctl_mport_mosi;
195:
196:                        mport3_miso : in typ_mctl_mport_miso;
197:                        mport3_mosi : out typ_mctl_mport_mosi
198:                );
199:        end component;
```

```
200:
201:         component cpt_mctl is
202:                 generic (
203:                         INCLUDE_MCTL_CHIPSCOPE  : string;
204:                         C3_P0_MASK_SIZE         : integer;
205:                         C3_P0_DATA_PORT_SIZE    : integer;
206:                         C3_P1_MASK_SIZE         : integer;
207:                         C3_P1_DATA_PORT_SIZE    : integer;
208:
209:                         C3_MEMCLK_PERIOD        : integer;
210:                         C3_RST_ACT_LOW          : integer;
211:                         C3_INPUT_CLK_TYPE       : string;
212:                         DEBUG_EN                : integer;
213:
214:                         C3_CALIB_SOFT_IP        : string;
215:                         C3_SIMULATION           : string;
216:                         C3_MEM_ADDR_ORDER       : string;
217:                         C3_NUM_DQ_PINS          : integer;
218:                         C3_MEM_ADDR_WIDTH       : integer;
219:                         C3_MEM_BANKADDR_WIDTH   : integer
220:                 );
221:                 port (
222:                         mcb3_dram_dq            : inout std_logic_vector(C3_NUM_DQ_PINS-1 downto 0);
223:                         mcb3_dram_dqs           : inout std_logic;
224:                         mcb3_dram_udqs          : inout std_logic;
225:                         mcb3_dram_a             : out std_logic_vector(C3_MEM_ADDR_WIDTH-1 downto 0);
226:                         mcb3_dram_ba            : out std_logic_vector(C3_MEM_BANKADDR_WIDTH-1 downto 0);
227:                         mcb3_dram_ras_n         : out std_logic;
228:                         mcb3_dram_cas_n         : out std_logic;
229:                         mcb3_dram_we_n          : out std_logic;
230:                         mcb3_dram_cke           : out std_logic;
231:                         mcb3_dram_dm            : out std_logic;
232:                         mcb3_dram_ck            : out std_logic;
233:                         mcb3_dram_udm           : out std_logic;
234:                         mcb3_dram_ck_n          : out std_logic;
235:
236:                         mcb3_rzq                : inout std_logic;
237:                         c3_sys_clk              : in std_logic;
238:                         c3_sys_rst_i            : in std_logic;
239:                         c3_calib_done           : out std_logic;
240:                         c3_clk0                 : out std_logic;
241:                         c3_rst0                 : out std_logic;
242:
243:                         clk_108                 : out std_logic;
244:                         clk_108_n               : out std_logic;
245:
246:                         c3_p0_cmd_clk           : in std_logic;
247:                         c3_p0_cmd_en            : in std_logic;
248:                         c3_p0_cmd_instr         : in std_logic_vector(2 downto 0);
249:                         c3_p0_cmd_bl            : in std_logic_vector(5 downto 0);
250:                         c3_p0_cmd_byte_addr     : in std_logic_vector(29 downto 0);
251:                         c3_p0_cmd_empty         : out std_logic;
252:                         c3_p0_cmd_full          : out std_logic;
253:                         c3_p0_wr_clk            : in std_logic;
254:                         c3_p0_wr_en             : in std_logic;
255:                         c3_p0_wr_mask           : in std_logic_vector(C3_P0_MASK_SIZE - 1 downto 0);
256:                         c3_p0_wr_data           : in std_logic_vector(C3_P0_DATA_PORT_SIZE - 1 downto 0);
257:                         c3_p0_wr_full           : out std_logic;
258:                         c3_p0_wr_empty          : out std_logic;
259:                         c3_p0_wr_count          : out std_logic_vector(6 downto 0);
260:                         c3_p0_wr_underrun       : out std_logic;
261:                         c3_p0_wr_error          : out std_logic;
262:                         c3_p0_rd_clk            : in std_logic;
263:                         c3_p0_rd_en             : in std_logic;
264:                         c3_p0_rd_data           : out std_logic_vector(C3_P0_DATA_PORT_SIZE - 1 downto 0);
265:                         c3_p0_rd_full           : out std_logic;
266:                         c3_p0_rd_empty          : out std_logic;
267:                         c3_p0_rd_count          : out std_logic_vector(6 downto 0);
268:                         c3_p0_rd_overflow       : out std_logic;
269:                         c3_p0_rd_error          : out std_logic;
270:                         c3_p1_cmd_clk           : in std_logic;
271:                         c3_p1_cmd_en            : in std_logic;
272:                         c3_p1_cmd_instr         : in std_logic_vector(2 downto 0);
273:                         c3_p1_cmd_bl            : in std_logic_vector(5 downto 0);
274:                         c3_p1_cmd_byte_addr     : in std_logic_vector(29 downto 0);
275:                         c3_p1_cmd_empty         : out std_logic;
276:                         c3_p1_cmd_full          : out std_logic;
277:                         c3_p1_wr_clk            : in std_logic;
278:                         c3_p1_wr_en             : in std_logic;
279:                         c3_p1_wr_mask           : in std_logic_vector(C3_P1_MASK_SIZE - 1 downto 0);
280:                         c3_p1_wr_data           : in std_logic_vector(C3_P1_DATA_PORT_SIZE - 1 downto 0);
281:                         c3_p1_wr_full           : out std_logic;
282:                         c3_p1_wr_empty          : out std_logic;
283:                         c3_p1_wr_count          : out std_logic_vector(6 downto 0);
284:                         c3_p1_wr_underrun       : out std_logic;
285:                         c3_p1_wr_error          : out std_logic;
286:                         c3_p1_rd_clk            : in std_logic;
287:                         c3_p1_rd_en             : in std_logic;
288:                         c3_p1_rd_data           : out std_logic_vector(C3_P1_DATA_PORT_SIZE - 1 downto 0);
289:                         c3_p1_rd_full           : out std_logic;
290:                         c3_p1_rd_empty          : out std_logic;
291:                         c3_p1_rd_count          : out std_logic_vector(6 downto 0);
292:                         c3_p1_rd_overflow       : out std_logic;
293:                         c3_p1_rd_error          : out std_logic;
294:                         c3_p2_cmd_clk           : in std_logic;
295:                         c3_p2_cmd_en            : in std_logic;
296:                         c3_p2_cmd_instr         : in std_logic_vector(2 downto 0);
297:                         c3_p2_cmd_bl            : in std_logic_vector(5 downto 0);
298:                         c3_p2_cmd_byte_addr     : in std_logic_vector(29 downto 0);
299:                         c3_p2_cmd_empty         : out std_logic;
```

```
300:                        c3_p2_cmd_full         : out std_logic;
301:                        c3_p2_wr_clk           : in std_logic;
302:                        c3_p2_wr_en            : in std_logic;
303:                        c3_p2_wr_mask          : in std_logic_vector(3 downto 0);
304:                        c3_p2_wr_data          : in std_logic_vector(31 downto 0);
305:                        c3_p2_wr_full          : out std_logic;
306:                        c3_p2_wr_empty         : out std_logic;
307:                        c3_p2_wr_count         : out std_logic_vector(6 downto 0);
308:                        c3_p2_wr_underrun      : out std_logic;
309:                        c3_p2_wr_error         : out std_logic;
310:                        c3_p2_rd_clk           : in std_logic;
311:                        c3_p2_rd_en            : in std_logic;
312:                        c3_p2_rd_data          : out std_logic_vector(31 downto 0);
313:                        c3_p2_rd_full          : out std_logic;
314:                        c3_p2_rd_empty         : out std_logic;
315:                        c3_p2_rd_count         : out std_logic_vector(6 downto 0);
316:                        c3_p2_rd_overflow      : out std_logic;
317:                        c3_p2_rd_error         : out std_logic;
318:                        c3_p3_cmd_clk          : in std_logic;
319:                        c3_p3_cmd_en           : in std_logic;
320:                        c3_p3_cmd_instr        : in std_logic_vector(2 downto 0);
321:                        c3_p3_cmd_bl           : in std_logic_vector(5 downto 0);
322:                        c3_p3_cmd_byte_addr    : in std_logic_vector(29 downto 0);
323:                        c3_p3_cmd_empty        : out std_logic;
324:                        c3_p3_cmd_full         : out std_logic;
325:                        c3_p3_wr_clk           : in std_logic;
326:                        c3_p3_wr_en            : in std_logic;
327:                        c3_p3_wr_mask          : in std_logic_vector(3 downto 0);
328:                        c3_p3_wr_data          : in std_logic_vector(31 downto 0);
329:                        c3_p3_wr_full          : out std_logic;
330:                        c3_p3_wr_empty         : out std_logic;
331:                        c3_p3_wr_count         : out std_logic_vector(6 downto 0);
332:                        c3_p3_wr_underrun      : out std_logic;
333:                        c3_p3_wr_error         : out std_logic;
334:                        c3_p3_rd_clk           : in std_logic;
335:                        c3_p3_rd_en            : in std_logic;
336:                        c3_p3_rd_data          : out std_logic_vector(31 downto 0);
337:                        c3_p3_rd_full          : out std_logic;
338:                        c3_p3_rd_empty         : out std_logic;
339:                        c3_p3_rd_count         : out std_logic_vector(6 downto 0);
340:                        c3_p3_rd_overflow      : out std_logic;
341:                        c3_p3_rd_error         : out std_logic
342:                );
343:        end component;
344: end pkg_mctl;
345:
346: package body pkg_mctl is
347:
348: end pkg_mctl;
```

```
 1:
 2:
 3: -- Basic Microblaze IO bus counter device
 4:
 5:
 6: library ieee;
 7: use ieee.std_logic_1164.all;
 8: use ieee.numeric_std.all;
 9:
10:
11: library mcu;
12: use mcu.pkg_mcu.all;
13:
14: library util;
15: use util.pkg_util.all;
16:
17:
18: entity cpt_counter is
19:
20:         generic (
21:                 DEVICE_ID : std_logic_vector(31 downto 0);
22:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0);
23:                 MCU_FREQUENCY : integer
24:         );
25:
26:         port (
27:                 i_clk : in std_logic;
28:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
29:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso
30:         );
31:
32: end cpt_counter;
33:
34: architecture Behavioral of cpt_counter is
35:
36:         signal count : integer := 0;
37:
38: begin
39:
40:         o_mcu_iobus_miso.ready <= '1';
41:         o_mcu_iobus_miso.read_data <= std_logic_vector(to_unsigned(count, 32));
42:
43:         process(i_clk)
44:         begin
45:                 if ( rising_edge(i_clk) ) then
46:                         count <= count + 1;
47:                 end if;
48:         end process;
49:
50:
51:
52: end Behavioral;
53:
```

```
    1:
    2: library ieee;
    3: use ieee.std_logic_1164.all;
    4: use ieee.std_logic_arith.all;
    5:
    6: library mcu;
    7: use mcu.pkg_mcu.all;
    8:
    9: entity cpt_gpio is
   10:
   11:         generic (
   12:                 DEVICE_ID : std_logic_vector(31 downto 0);
   13:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0);
   14:                 N_GPIOS : integer
   15:         );
   16:
   17:         port (
   18:                 i_clk : in std_logic;
   19:                 i_reset : in std_logic;
   20:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
   21:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso;
   22:                 i_gpi : in typ_mcu_word_array;
   23:                 o_gpo : out typ_mcu_word_array
   24:         );
   25:
   26: end cpt_gpio;
   27:
   28:
   29: architecture Behavioral of cpt_gpio is
   30:
   31: begin
   32:
   33:         process(i_clk)
   34:         begin
   35:                 if ( rising_edge(i_clk) ) then
   36:                         o_mcu_iobus_miso.ready <= i_mcu_iobus_mosi.addr_strobe;
   37:                 end if;
   38:         end process;
   39:
   40:
   41:         process(i_clk)
   42:         begin
   43:                 if ( rising_edge(i_clk) ) then
   44:                         if ( i_reset = '1' ) then
   45:                                 for i in 0 to N_GPIOS-1 loop
   46:                                         o_gpo(i) <= (others => '0');
   47:                                 end loop;
   48:
   49:                         elsif ( (i_mcu_iobus_mosi.address and DEVICE_ID_MASK) = (DEVICE_ID and DEVICE_ID_MASK) ) then
   50:
   51:                                 for  i in 0 to N_GPIOS-1 loop
   52:
   53:                                         if ( i_mcu_iobus_mosi.read_strobe = '1' and i_mcu_iobus_mosi.address(25 downto 2) = conv_std_logic_vector(i,
24) ) then
   54:                                                 o_mcu_iobus_miso.read_data <= i_gpi(i);
   55:                                         end if;
   56:
   57:                                         if ( i_mcu_iobus_mosi.write_strobe = '1' and i_mcu_iobus_mosi.address(25 downto 2) = conv_std_logic_vector(i,
 24) ) then
   58:                                                 o_gpo(i) <= i_mcu_iobus_mosi.write_data;
   59:                                         end if;
   60:
   61:                                 end loop;
   62:
   63:                         end if;
   64:                 end if;
   65:         end process;
   66:
   67:
   68: end Behavioral;
   69:
```

```
    1: --
    2: -- RAM/IOBus link
    3: -- Provides an interface for the Microblaze MCU to read and write external memory via the IOBus
    4: --
    5:
    6: library ieee;
    7: use ieee.std_logic_1164.all;
    8:
    9: library mctl;
   10: use mctl.pkg_mctl.all;
   11:
   12: library mcu;
   13: use mcu.pkg_mcu.all;
   14:
   15: entity cpt_iobus_mport is
   16:         generic (
   17:                 DEVICE_ID : std_logic_vector(31 downto 0);
   18:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0)
   19:         );
   20:         port (
   21:                 i_clk : in std_logic;
   22:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
   23:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso;
   24:                 i_mctl_mport_mosi : in typ_mctl_mport_mosi;
   25:                 o_mctl_mport_miso : out typ_mctl_mport_miso
   26:         );
   27: end cpt_iobus_mport;
   28:
   29: architecture Behavioral of cpt_iobus_mport is
   30:         signal addr_latch : std_logic_vector(31 downto 0);
   31:         signal wr_data_latch : std_logic_vector(31 downto 0);
   32:
   33:         constant RAMSTATE_IDLE : integer := 16#0000#;
   34:         constant RAMSTATE_READ : integer := 16#0010#;
   35:         constant RAMSTATE_WRITE : integer := 16#0020#;
   36:
   37:         signal ramstate : integer := RAMSTATE_IDLE;
   38: begin
   39:
   40:         -- Send input clock to all other clock signals
   41:         o_mctl_mport_miso.cmd.clk <= i_clk;
   42:         o_mctl_mport_miso.wr.clk <= i_clk;
   43:         o_mctl_mport_miso.rd.clk <= i_clk;
   44:
   45:         process (i_clk)
   46:         begin
   47:                 if rising_edge(i_clk) then
   48:
   49:                         -- IO Bus
   50:                         o_mcu_iobus_miso.read_data <= (others => '0');
   51:                         o_mcu_iobus_miso.ready <= '0';
   52:
   53:                         -- Command signal defaults
   54:                         o_mctl_mport_miso.cmd.en <= '0';
   55:                         o_mctl_mport_miso.cmd.instr <= "000";
   56:                         o_mctl_mport_miso.cmd.bl <= "000000";
   57:                         o_mctl_mport_miso.cmd.byte_addr <= (others => '0');
   58:
   59:                         -- Write signal defaults
   60:                         o_mctl_mport_miso.wr.en <= '0';
   61:                         o_mctl_mport_miso.wr.mask <= "0000";
   62:
   63:                         -- Read signal defaults
   64:                         o_mctl_mport_miso.rd.en <= '0';
   65:
   66:                         case ramstate is
   67:                                 when RAMSTATE_IDLE+0 =>
   68:                                         if ( (i_mcu_iobus_mosi.address and DEVICE_ID_MASK) = (DEVICE_ID and DEVICE_ID_MASK) ) then
   69:
   70:                                                 if ( i_mcu_iobus_mosi.Read_Strobe = '1' ) then
   71:                                                         addr_latch <= i_mcu_iobus_mosi.address;
   72:                                                         ramstate <= RAMSTATE_READ;
   73:                                                 end if;
   74:
   75:                                                 if ( i_mcu_iobus_mosi.Write_Strobe = '1' ) then
   76:                                                         addr_latch <= i_mcu_iobus_mosi.address;
   77:                                                         wr_data_latch <= i_mcu_iobus_mosi.write_data;
   78:                                                         ramstate <= RAMSTATE_WRITE;
   79:                                                 end if;
   80:
   81:                                         end if;
   82:
   83:                                 when RAMSTATE_READ+0 =>
   84:                                         if ( i_mctl_mport_mosi.cmd.full /= '1' ) then
   85:                                                 o_mctl_mport_miso.cmd.en <= '1';
   86:                                                 o_mctl_mport_miso.cmd.instr <= "011";
   87:                                                 o_mctl_mport_miso.cmd.bl <= "000000";
   88:                                                 o_mctl_mport_miso.cmd.byte_addr <= "0000" & addr_latch(25 downto 0);
   89:                                                 ramstate <= ramstate + 1;
   90:                                         end if;
   91:
   92:                                 when RAMSTATE_READ+1 =>
   93:                                         if ( i_mctl_mport_mosi.rd.empty = '0' ) then
   94:                                                 --rd.en <= '1';
   95:                                                 --rd.data_latch <= rd.data;
   96:                                                 ramstate <= ramstate + 1;
   97:                                         end if;
   98:
   99:                                 when RAMSTATE_READ+2 =>
  100:                                         if ( i_mctl_mport_mosi.rd.empty = '0' ) then
```

```
101:                                            o_mctl_mport_miso.rd.en <= '1';
102:                                            o_mcu_iobus_miso.read_data <= i_mctl_mport_mosi.rd.data;
103:                                            o_mcu_iobus_miso.ready <= '1';
104:                                            ramstate <= RAMSTATE_IDLE;
105:                                        end if;
106:
107:                                    when RAMSTATE_WRITE+0 =>
108:                                        if ( i_mctl_mport_mosi.wr.full /= '1' ) then
109:                                            --wr.en <= '1';
110:                                            o_mctl_mport_miso.wr.data <= wr_data_latch;
111:                                            ramstate <= ramstate + 1;
112:                                        end if;
113:
114:                                    when RAMSTATE_WRITE+1 =>
115:                                        if ( i_mctl_mport_mosi.wr.full /= '1' ) then
116:                                            o_mctl_mport_miso.wr.en <= '1';
117:                                            o_mctl_mport_miso.wr.data <= wr_data_latch;
118:                                            ramstate <= ramstate + 1;
119:                                        end if;
120:
121:                                    when RAMSTATE_WRITE+2 =>
122:                                        if ( i_mctl_mport_mosi.cmd.full /= '1' ) then
123:                                            o_mctl_mport_miso.cmd.en <= '1';
124:                                            o_mctl_mport_miso.cmd.instr <= "010";
125:                                            o_mctl_mport_miso.cmd.bl <= "000000";
126:                                            o_mctl_mport_miso.cmd.byte_addr <= "0000" & addr_latch(25 downto 0);
127:                                            o_mcu_iobus_miso.ready <= '1';
128:                                            ramstate <= RAMSTATE_IDLE;
129:                                        end if;
130:
131:                                    when others =>
132:                                        ramstate <= RAMSTATE_IDLE;
133:
134:                            end case;
135:                    end if;
136:            end process;
137: end Behavioral;
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4: use IEEE.std_logic_arith.ALL;
  5: use ieee.std_logic_unsigned.all;
  6:
  7: library unisim;
  8: use unisim.vcomponents.all;
  9:
 10: library mctl;
 11: use mctl.pkg_mctl.all;
 12:
 13: library cctl;
 14: use cctl.pkg_cctl.all;
 15: use cctl.pkg_ovm.all;
 16:
 17: --library wifi;
 18: --use wifi.pkg_wifi.all;
 19:
 20: library mcu;
 21: use mcu.pkg_mcu.all;
 22:
 23: library fast_uart;
 24: use fast_uart.pkg_fast_uart.all;
 25:
 26: library util;
 27: use util.pkg_util.all;
 28:
 29:
 30: entity cpt_mcu is
 31:         generic (
 32:                 INCLUDE_IOBUS_MPORT : string := "TRUE";
 33:                 INCLUDE_SCCB : string := "TRUE"
 34:         );
 35:         port (
 36:                 i_clk : in std_logic;
 37:                 i_reset : in std_logic;
 38:
 39:                 o_debug_output_enable : out std_logic;
 40:                 o_debug_src : out integer range 0 to 15;
 41:
 42:                 i_mctl_mport_mosi : in typ_mctl_mport_mosi;
 43:                 o_mctl_mport_miso : out typ_mctl_mport_miso;
 44:
 45:                 io_ovm0_sccb_bidir : inout typ_ovm_sccb_bidir;
 46:                 o_ovm0_sccb_mosi : out typ_ovm_sccb_mosi;
 47:                 io_ovm1_sccb_bidir : inout typ_ovm_sccb_bidir;
 48:                 o_ovm1_sccb_mosi : out typ_ovm_sccb_mosi;
 49:                 io_ovm2_sccb_bidir : inout typ_ovm_sccb_bidir;
 50:                 o_ovm2_sccb_mosi : out typ_ovm_sccb_mosi;
 51:                 io_ovm3_sccb_bidir : inout typ_ovm_sccb_bidir;
 52:                 o_ovm3_sccb_mosi : out typ_ovm_sccb_mosi;
 53:
 54:                 i_uart_rx : in std_logic := '1';
 55:                 o_uart_tx : out std_logic := '1';
 56:
 57:                 o_wifi_txd : inout std_logic;
 58:                 io_wifi_rxd : inout std_logic;
 59:                 o_wifi_rst : out std_logic;
 60:                 io_wifi_gpio0 : inout std_logic;
 61:                 io_wifi_gpio2 : inout std_logic;
 62:                 o_wifi_ch_pd : out std_logic;
 63:
 64:
 65:                 i_gpi : in typ_mcu_word_array;
 66:                 o_gpo : out typ_mcu_word_array;
 67:
 68:                 i_intc_interrupt : in std_logic_vector(7 downto 0);
 69:                 o_intc_irq : out std_logic_vector(31 downto 0)
 70:         );
 71: end cpt_mcu;
 72:
 73: architecture Behavioral of cpt_mcu is
 74:
 75:         signal gpi : typ_mcu_word_array;
 76:         signal gpo : typ_mcu_word_array;
 77:
 78:         signal iobus_device_id : std_logic_vector(31 downto 0) := x"00000000";
 79:
 80:         signal iobus_mosi : typ_mcu_iobus_mosi;
 81:         signal iobus_miso : typ_mcu_iobus_miso;
 82:
 83:         constant IOBUS_DEVICE_ID_MASK : std_logic_vector(31 downto 0) := x"FC000000";
 84:
 85:         -- These set the base addresses for IO bus devices
 86:         constant NULL_DEVICE_ID         : std_logic_vector(31 downto 0) := x"00000000";
 87:         constant MPORT_DEVICE_ID        : std_logic_vector(31 downto 0) := x"C0000000";
 88:         constant TIMER_DEVICE_ID        : std_logic_vector(31 downto 0) := x"D0000000";
 89:         constant COUNTER_DEVICE_ID : std_logic_vector(31 downto 0) := x"D4000000";
 90:         constant GPIO_DEVICE_ID         : std_logic_vector(31 downto 0) := x"D8000000";
 91:         constant OVM0_DEVICE_ID         : std_logic_vector(31 downto 0) := x"E0000000";
 92:         constant OVM1_DEVICE_ID         : std_logic_vector(31 downto 0) := x"E4000000";
 93:         constant OVM2_DEVICE_ID         : std_logic_vector(31 downto 0) := x"E8000000";
 94:         constant OVM3_DEVICE_ID         : std_logic_vector(31 downto 0) := x"EC000000";
 95:         constant UART_TX_DEVICE_ID      : std_logic_vector(31 downto 0) := x"F0000000";
 96:         constant UART_RX_DEVICE_ID      : std_logic_vector(31 downto 0) := x"F4000000";
 97:         constant WIFI_TX_DEVICE_ID      : std_logic_vector(31 downto 0) := x"F8000000";
 98:         constant WIFI_RX_DEVICE_ID      : std_logic_vector(31 downto 0) := x"FC000000";
 99:
100:         signal mport_iobus_miso : typ_mcu_iobus_miso;
```

```
101:         signal timer_iobus_miso : typ_mcu_iobus_miso;
102:         signal counter_iobus_miso : typ_mcu_iobus_miso;
103:         signal gpio_iobus_miso : typ_mcu_iobus_miso;
104:         signal ovm0_iobus_miso : typ_mcu_iobus_miso;
105:         signal ovm1_iobus_miso : typ_mcu_iobus_miso;
106:         signal ovm2_iobus_miso : typ_mcu_iobus_miso;
107:         signal ovm3_iobus_miso : typ_mcu_iobus_miso;
108:         signal uart_tx_iobus_miso : typ_mcu_iobus_miso;
109:         signal uart_rx_iobus_miso : typ_mcu_iobus_miso;
110:         signal wifi_tx_iobus_miso : typ_mcu_iobus_miso;
111:         signal wifi_rx_iobus_miso : typ_mcu_iobus_miso;
112:
113:
114:         signal wifi_rxd_oe_n : std_logic;
115:         signal wifi_rxd_o : std_logic;
116:         signal wifi_rxd_i : std_logic;
117:
118:         signal wifi_gpio_oe_n : std_logic_vector(2 downto 0);
119:         signal wifi_gpo : std_logic_vector(2 downto 0);
120:         signal wifi_gpi : std_logic_vector(2 downto 0);
121:
122:         signal debug_output_enable : std_logic;
123:         signal debug_src : integer range 0 to 15;
124:
125:         signal debug_ovm0_enable : std_logic;
126:         signal ovm0_enable : std_logic;
127:         signal ovm1_enable : std_logic;
128:         signal ovm2_enable : std_logic;
129:         signal ovm3_enable : std_logic;
130:
131:         signal ovm_scl_clk_div : integer;
132:         signal ovm_xvclk_div : integer;
133:         signal ovm_dev_addr : std_logic_vector(6 downto 0);
134:
135:
136:         signal uart_txd : std_logic;
137:         signal uart_rxd : std_logic;
138:
139:         signal uart_tx_full : std_logic;
140:         signal uart_tx_empty : std_logic;
141:         signal uart_rx_full : std_logic;
142:         signal uart_rx_empty : std_logic;
143:         signal uart_rx_empty_n : std_logic;
144:
145:         signal uart_rx_src : std_logic;
146:         signal uart_tx_src : std_logic;
147:
148:         signal uart_baud_div : integer;
149:
150:
151:         signal wifi_txd : std_logic;
152:         signal wifi_rxd : std_logic;
153:
154:         signal wifi_tx_full : std_logic;
155:         signal wifi_tx_empty : std_logic;
156:         signal wifi_rx_full : std_logic;
157:         signal wifi_rx_empty : std_logic;
158:         signal wifi_rx_empty_n : std_logic;
159:
160:         signal wifi_rx_src : std_logic;
161:         signal wifi_tx_src : std_logic;
162:
163:
164:         signal wifi_enable : std_logic_vector(1 downto 0);
165:         signal wifi_baud_div : integer;
166:
167:
168:
169:         constant MCU_FREQUENCY : integer := 108000000; -- 108MHz
170:
171:
172: begin
173:
174:
175:         microblaze : cpt_microblaze
176:         port map (
177:                 Clk => i_clk,
178:                 Reset => i_reset,
179:                 IO_Addr_Strobe => iobus_mosi.addr_strobe,
180:                 IO_Read_Strobe => iobus_mosi.read_strobe,
181:                 IO_Write_Strobe => iobus_mosi.write_strobe,
182:                 IO_Address => iobus_mosi.address,
183:                 IO_Byte_Enable => iobus_mosi.byte_enable,
184:                 IO_Write_Data => iobus_mosi.write_data,
185:                 IO_Read_Data => iobus_miso.read_data,
186:                 IO_Ready => iobus_miso.ready,
187:                 UART_Rx => '1',
188:                 UART_Tx => open,
189:                 UART_Interrupt => open,
190:                 GPO1 => open,
191:                 GPO2 => open,
192:                 GPO3 => open,
193:                 GPO4 => open,
194:                 GPI1 => (others => '0'),
195:                 GPI1_Interrupt => open,
196:                 GPI2 => (others => '0'),
197:                 GPI2_Interrupt => open,
198:                 GPI3 => (others => '0'),
199:                 GPI3_Interrupt => open,
200:                 GPI4 => (others => '0'),
```

```
201:                GPI4_Interrupt => open,
202:                INTC_Interrupt(0) => i_intc_interrupt(0),
203:                INTC_Interrupt(1) => uart_rx_empty_n,
204:                INTC_Interrupt(2) => wifi_rx_empty_n,
205:                INTC_Interrupt(7 downto 3) => (others => '0'),
206:                INTC_IRQ => open
207:        );
208:
209:
210:
211:
212:        process(i_clk)
213:        begin
214:                if ( rising_edge(i_clk) ) then
215:                        if ( iobus_mosi.addr_strobe = '1' and iobus_mosi.address(31 downto 30) = "11" ) then
216:                                iobus_device_id <= iobus_mosi.address and IOBUS_DEVICE_ID_MASK;
217:                        elsif ( iobus_miso.ready = '1' ) then
218:                                iobus_device_id <= NULL_DEVICE_ID and IOBUS_DEVICE_ID_MASK;
219:                        end if;
220:                end if;
221:        end process;
222:
223:
224:
225:
226:        with iobus_device_id select
227:                iobus_miso <=   mport_iobus_miso when MPORT_DEVICE_ID,
228:                                                        timer_iobus_miso when TIMER_DEVICE_ID,
229:                                                        counter_iobus_miso when COUNTER_DEVICE_ID,
230:                                                        gpio_iobus_miso when GPIO_DEVICE_ID,
231:                                                        ovm0_iobus_miso when OVM0_DEVICE_ID,
232:                                                        ovm1_iobus_miso when OVM1_DEVICE_ID,
233:                                                        ovm2_iobus_miso when OVM2_DEVICE_ID,
234:                                                        ovm3_iobus_miso when OVM3_DEVICE_ID,
235:                                                        uart_tx_iobus_miso when UART_TX_DEVICE_ID,
236:                                                        uart_rx_iobus_miso when UART_RX_DEVICE_ID,
237:                                                        wifi_tx_iobus_miso when WIFI_TX_DEVICE_ID,
238:                                                        wifi_rx_iobus_miso when WIFI_RX_DEVICE_ID,
239:                                                        init_mcu_iobus_miso when others;
240:
241:
242:
243:
244:
245:
246:
247:        incl_iobus_mport :
248:        if ( INCLUDE_IOBUS_MPORT = "TRUE" ) generate
249:
250:                iobus_mport : cpt_iobus_mport
251:                generic map (
252:                        DEVICE_ID => MPORT_DEVICE_ID,
253:                        DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
254:                )
255:                port map (
256:                        i_clk => i_clk,
257:                        i_mcu_iobus_mosi => iobus_mosi,
258:                        o_mcu_iobus_miso => mport_iobus_miso,
259:                        i_mctl_mport_mosi => i_mctl_mport_mosi,
260:                        o_mctl_mport_miso =>  o_mctl_mport_miso
261:                );
262:
263:        end generate incl_iobus_mport;
264:
265:
266:
267:
268:        timer : cpt_timer
269:        generic map (
270:                DEVICE_ID => TIMER_DEVICE_ID,
271:                DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK,
272:                MCU_FREQUENCY => MCU_FREQUENCY
273:        )
274:        port map (
275:                i_clk => i_clk,
276:                i_mcu_iobus_mosi => iobus_mosi,
277:                o_mcu_iobus_miso => timer_iobus_miso
278:        );
279:
280:
281:        counter : cpt_counter
282:        generic map (
283:                DEVICE_ID => COUNTER_DEVICE_ID,
284:                DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK,
285:                MCU_FREQUENCY => MCU_FREQUENCY
286:        )
287:        port map (
288:                i_clk => i_clk,
289:                i_mcu_iobus_mosi => iobus_mosi,
290:                o_mcu_iobus_miso => counter_iobus_miso
291:        );
292:
293:
294:        --gpi <= gpo;
295:
296:        gpi(0 to 16#80#-1) <= i_gpi(0 to 16#80#-1);
297:        o_gpo(0 to 16#80#-1) <= gpo(0 to 16#80#-1);
298:
299:
300:        gpio : cpt_gpio
```

```
301:         generic map (
302:                 DEVICE_ID => GPIO_DEVICE_ID,
303:                 DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK,
304:                 N_GPIOS => N_GPIOS
305:         )
306:         port map (
307:                 i_clk => i_clk,
308:                 i_reset => i_reset,
309:                 i_mcu_iobus_mosi => iobus_mosi,
310:                 o_mcu_iobus_miso => gpio_iobus_miso,
311:                 i_gpi => gpi,
312:                 o_gpo => gpo
313:         );
314:
315:
316:         uart_baud_div <= conv_integer(gpo(GPIO_UART_BAUD_DIV));
317:
318:
319:         gpi(GPIO_UART_STATUS) <= (
320:                 0 => uart_rx_empty,
321:                 1 => uart_rx_full,
322:                 2 => i_uart_rx,
323:                 4 => uart_tx_empty,
324:                 5 => uart_tx_full,
325:                 others => '0'
326:         );
327:
328:
329:         fast_uart_tx : cpt_fast_uart_tx
330:         generic map (
331:                 DEVICE_ID => UART_TX_DEVICE_ID,
332:                 DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
333:         )
334:         port map (
335:                 clk => i_clk,
336:                 reset => i_reset,
337:                 enable => '1',
338:                 baud_div => uart_baud_div,
339:                 i_mcu_iobus_mosi => iobus_mosi,
340:                 o_mcu_iobus_miso => uart_tx_iobus_miso,
341:                 empty => uart_tx_empty,
342:                 full => uart_tx_full,
343:                 txd => uart_txd
344:         );
345:
346:
347:
348:
349:
350:         uart_tx_src <= gpo(GPIO_UART_TX_SRC)(0);
351:
352:         with uart_tx_src select o_uart_tx <=
353:                 uart_txd when '0',
354:                 wifi_rxd_i when '1';
355:
356:
357:         wifi_tx_src <= gpo(GPIO_WIFI_TX_SRC)(0);
358:
359:         with wifi_tx_src select o_wifi_txd <=
360:                 wifi_txd when '0',
361:                 i_uart_rx when '1';
362:
363:
364:
365:
366:         uart_rx_src <= gpo(GPIO_UART_RX_SRC)(0);
367:
368:         with uart_rx_src select uart_rxd <=
369:                 i_uart_rx when '0',
370:                 uart_txd when '1'; --loopback
371:
372:
373:         uart_rx_empty_n <= not uart_rx_empty;
374:
375:         fast_uart_rx : cpt_fast_uart_rx
376:         generic map (
377:                 DEVICE_ID => UART_RX_DEVICE_ID,
378:                 DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
379:         )
380:         port map (
381:                 clk => i_clk,
382:                 reset => i_reset,
383:                 enable => '1',
384:                 baud_div => uart_baud_div,
385:                 i_mcu_iobus_mosi => iobus_mosi,
386:                 o_mcu_iobus_miso => uart_rx_iobus_miso,
387:                 full => uart_rx_full,
388:                 empty => uart_rx_empty,
389:                 rxd => uart_rxd
390:         );
391:
392:
393:
394:         wifi_baud_div <= conv_integer(gpo(GPIO_WIFI_BAUD_DIV));
395:
396:
397:         gpi(GPIO_WIFI_STATUS) <= (
398:                 0 => wifi_rx_empty,
399:                 1 => wifi_rx_full,
400:                 2 => wifi_rxd_i,
```

```
401:                4 => wifi_tx_empty,
402:                5 => wifi_tx_full,
403:                others => '0'
404:        );
405:
406:
407:        wifi_tx : cpt_fast_uart_tx
408:        generic map (
409:                DEVICE_ID => WIFI_TX_DEVICE_ID,
410:                DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
411:        )
412:        port map (
413:                clk => i_clk,
414:                reset => i_reset,
415:                enable => '1',
416:                baud_div => wifi_baud_div,
417:                i_mcu_iobus_mosi => iobus_mosi,
418:                o_mcu_iobus_miso => wifi_tx_iobus_miso,
419:                empty => wifi_tx_empty,
420:                full => wifi_tx_full,
421:                txd => wifi_txd
422:        );
423:
424:
425:        wifi_enable <= gpo(GPIO_WIFI_ENABLE)(1 downto 0);
426:
427:        --o_wifi_txd <= wifi_txd;
428:        o_wifi_rst <= wifi_enable(0);
429:        o_wifi_ch_pd <= wifi_enable(1);
430:
431:
432:
433:
434:        wifi_rxd_oe_n <= not gpo(GPIO_WIFI_RXD_OUTPUT_ENABLE)(0);
435:        wifi_rxd_o <= gpo(GPIO_WIFI_RXD)(0);
436:        gpi(GPIO_WIFI_RXD)(0) <= wifi_rxd_i;
437:
438:        wifi_rxd_iobuf : iobuf
439:        port map (
440:                io => io_wifi_rxd,
441:                i => wifi_rxd_o,
442:                o => wifi_rxd_i,
443:                t => wifi_rxd_oe_n
444:        );
445:
446:
447:        wifi_gpio_oe_n <= not gpo(GPIO_WIFI_GPIO_OUTPUT_ENABLE)(2 downto 0);
448:        wifi_gpo <= gpo(GPIO_WIFI_GPIO)(2 downto 0);
449:        gpi(GPIO_WIFI_GPIO)(2 downto 0) <= wifi_gpi;
450:
451:
452:        wifi_gpio0_iobuf : iobuf
453:        port map (
454:                io => io_wifi_gpio0,
455:                i => wifi_gpo(0),
456:                o => wifi_gpi(0),
457:                t => wifi_gpio_oe_n(0)
458:        );
459:
460:        wifi_gpio2_iobuf : iobuf
461:        port map (
462:                io => io_wifi_gpio2,
463:                i => wifi_gpo(2),
464:                o => wifi_gpi(2),
465:                t => wifi_gpio_oe_n(2)
466:        );
467:
468:
469:        wifi_rx_src <= gpo(GPIO_WIFI_RX_SRC)(0);
470:
471:        with wifi_rx_src select wifi_rxd <=
472:                wifi_rxd_i when '0',
473:                wifi_txd when '1';      -- loopback
474:
475:
476:        wifi_rx_empty_n <= not wifi_rx_empty;
477:
478:        wifi_rx : cpt_fast_uart_rx
479:        generic map (
480:                DEVICE_ID => wifi_RX_DEVICE_ID,
481:                DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
482:        )
483:        port map (
484:                clk => i_clk,
485:                reset => i_reset,
486:                enable => '1',
487:                baud_div => wifi_baud_div,
488:                i_mcu_iobus_mosi => iobus_mosi,
489:                o_mcu_iobus_miso => wifi_rx_iobus_miso,
490:                full => wifi_rx_full,
491:                empty => wifi_rx_empty,
492:                rxd => wifi_rxd
493:        );
494:
495:
496:
497:
498:
499:        debug_output_enable <= gpo(GPIO_DEBUG_OUTPUT_ENABLE)(0);
500:        o_debug_output_enable <= debug_output_enable;
```

```
501:
502:         debug_src <= conv_integer(gpo(GPIO_DEBUG_SRC));
503:         o_debug_src <= debug_src;
504:
505:
506:         ovm0_enable <= gpo(GPIO_OVM_ENABLE)(0);
507:         debug_ovm0_enable <= '0' when debug_output_enable = '1' else ovm0_enable;
508:
509:         ovm1_enable <= gpo(GPIO_OVM_ENABLE)(1);
510:         ovm2_enable <= gpo(GPIO_OVM_ENABLE)(2);
511:         ovm3_enable <= gpo(GPIO_OVM_ENABLE)(3);
512:
513:         gpi(GPIO_OVM_ENABLE) <= gpo(GPIO_OVM_ENABLE);
514:
515:         ovm_dev_addr <= gpo(GPIO_OVM_DEV_ADDR)(6 downto 0);
516:
517:         ovm_xvclk_div <= conv_integer(gpo(GPIO_OVM_XVCLK_DIV));
518:
519:         ovm_scl_clk_div <= conv_integer(gpo(GPIO_OVM_SCL_CLK_DIV));
520:
521:
522:         incl_sccb :
523:         if ( INCLUDE_SCCB = "TRUE" ) generate
524:
525:
526:                 o_ovm0_sccb_mosi.pwdn <= not debug_ovm0_enable;
527:
528:                 ovm0_xvclk_clkout : cpt_clkout
529:                 port map (
530:                         i_enable => debug_ovm0_enable,
531:                         i_clk => i_clk,
532:                         i_clk_div => ovm_xvclk_div,
533:                         o_clk => o_ovm0_sccb_mosi.xvclk
534:                 );
535:
536:                 ovm0_iobus_sccb : cpt_iobus_sccb
537:                 generic map (
538:                         DEVICE_ID => OVM0_DEVICE_ID,
539:                         DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
540:                 )
541:                 port map (
542:                         i_clk => i_clk,
543:                         i_enable => debug_ovm0_enable,
544:                         i_iobus_mosi => iobus_mosi,
545:                         o_iobus_miso => ovm0_iobus_miso,
546:                         i_dev_addr => ovm_dev_addr,
547:                         i_scl_clk_div => ovm_scl_clk_div,
548:                         io_scl => io_ovm0_sccb_bidir.scl,
549:                         io_sda => io_ovm0_sccb_bidir.sda
550:                 );
551:
552:
553:
554:
555:                 o_ovm1_sccb_mosi.pwdn <= not ovm1_enable;
556:
557:                 ovm1_xvclk_clkout : cpt_clkout
558:                 port map (
559:                         i_enable => ovm1_enable,
560:                         i_clk => i_clk,
561:                         i_clk_div => ovm_xvclk_div,
562:                         o_clk => o_ovm1_sccb_mosi.xvclk
563:                 );
564:
565:                 ovm1_iobus_sccb : cpt_iobus_sccb
566:                 generic map (
567:                         DEVICE_ID => OVM1_DEVICE_ID,
568:                         DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
569:                 )
570:                 port map (
571:                         i_clk => i_clk,
572:                         i_enable => ovm1_enable,
573:                         i_iobus_mosi => iobus_mosi,
574:                         o_iobus_miso => ovm1_iobus_miso,
575:                         i_dev_addr => ovm_dev_addr,
576:                         i_scl_clk_div => ovm_scl_clk_div,
577:                         io_scl => io_ovm1_sccb_bidir.scl,
578:                         io_sda => io_ovm1_sccb_bidir.sda
579:                 );
580:
581:
582:                 o_ovm2_sccb_mosi.pwdn <= not ovm2_enable;
583:
584:                 ovm2_xvclk_clkout : cpt_clkout
585:                 port map (
586:                         i_enable => ovm2_enable,
587:                         i_clk => i_clk,
588:                         i_clk_div => ovm_xvclk_div,
589:                         o_clk => o_ovm2_sccb_mosi.xvclk
590:                 );
591:
592:                 ovm2_iobus_sccb : cpt_iobus_sccb
593:                 generic map (
594:                         DEVICE_ID => OVM2_DEVICE_ID,
595:                         DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
596:                 )
597:                 port map (
598:                         i_clk => i_clk,
599:                         i_enable => ovm2_enable,
600:                         i_iobus_mosi => iobus_mosi,
```

```
601:                        o_iobus_miso => ovm2_iobus_miso,
602:                        i_dev_addr => ovm_dev_addr,
603:                        i_scl_clk_div => ovm_scl_clk_div,
604:                        io_scl => io_ovm2_sccb_bidir.scl,
605:                        io_sda => io_ovm2_sccb_bidir.sda
606:                );
607:
608:
609:                o_ovm3_sccb_mosi.pwdn <= not ovm3_enable;
610:
611:                ovm3_xvclk_clkout : cpt_clkout
612:                port map (
613:                        i_enable => ovm3_enable,
614:                        i_clk => i_clk,
615:                        i_clk_div => ovm_xvclk_div,
616:                        o_clk => o_ovm3_sccb_mosi.xvclk
617:                );
618:
619:                ovm3_iobus_sccb : cpt_iobus_sccb
620:                generic map (
621:                        DEVICE_ID => OVM3_DEVICE_ID,
622:                        DEVICE_ID_MASK => IOBUS_DEVICE_ID_MASK
623:                )
624:                port map (
625:                        i_clk => i_clk,
626:                        i_enable => ovm3_enable,
627:                        i_iobus_mosi => iobus_mosi,
628:                        o_iobus_miso => ovm3_iobus_miso,
629:                        i_dev_addr => ovm_dev_addr,
630:                        i_scl_clk_div => ovm_scl_clk_div,
631:                        io_scl => io_ovm3_sccb_bidir.scl,
632:                        io_sda => io_ovm3_sccb_bidir.sda
633:                );
634:
635:
636:        end generate incl_sccb;
637:
638:
639:
640:
641: end Behavioral;
642:
```

```
  1:
  2:
  3: -- Basic Microblaze IO bus timer device
  4:
  5: -- This timer is implemented with a software-writable 32-bit register and a hardware counter.
  6: -- When a value is written to the timer register, the counter resets to zero and begins counting up.
  7: -- The IO bus Ready line is held low until the counter value is equal to the register value.
  8: -- Program execution is halted until Ready is set high.
  9:
 10: -- TODO: Are interrupts blocked while waiting for Ready? (probably...)
 11:
 12: library ieee;
 13: use ieee.std_logic_1164.all;
 14: use ieee.std_logic_arith.all;
 15: use ieee.std_logic_unsigned.all;
 16:
 17:
 18: library mcu;
 19: use mcu.pkg_mcu.all;
 20:
 21: library util;
 22: use util.pkg_util.all;
 23:
 24:
 25: entity cpt_timer is
 26:
 27:         generic (
 28:                 DEVICE_ID : std_logic_vector(31 downto 0);
 29:                 DEVICE_ID_MASK : std_logic_vector(31 downto 0);
 30:                 MCU_FREQUENCY : integer
 31:         );
 32:
 33:         port (
 34:                 i_clk : in std_logic;
 35:                 i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
 36:                 o_mcu_iobus_miso : out typ_mcu_iobus_miso
 37:         );
 38:
 39: end cpt_timer;
 40:
 41: architecture Behavioral of cpt_timer is
 42:
 43:         signal clk_pgate : std_logic := '0';
 44:         signal ready : std_logic := '1';
 45:         signal max_count : integer := 1;
 46:         signal enable : std_logic := '0';
 47:         signal enable_n : std_logic := '0';
 48:
 49: begin
 50:
 51:         o_mcu_iobus_miso.ready <= ready;
 52:
 53:
 54:         -- Timer register
 55:         process(i_clk)
 56:         begin
 57:                 if ( rising_edge(i_clk) and i_mcu_iobus_mosi.write_strobe = '1' ) then
 58:                         if ( (i_mcu_iobus_mosi.address and DEVICE_ID_MASK) = (DEVICE_ID and DEVICE_ID_MASK) ) then
 59:                                 max_count <= conv_integer(i_mcu_iobus_mosi.write_data);
 60:                         end if;
 61:                 end if;
 62:         end process;
 63:
 64:         -- Counter controller
 65:         -- Counter is enabled on writes to the timer register,
 66:         -- and disabled when Ready is asserted on timer expiry
 67:         process(i_clk)
 68:         begin
 69:                 if ( rising_edge(i_clk) ) then
 70:                         if ( i_mcu_iobus_mosi.write_strobe = '1' ) then
 71:                                 if ( (i_mcu_iobus_mosi.address and DEVICE_ID_MASK) = (DEVICE_ID and DEVICE_ID_MASK) ) then
 72:                                         enable <= '1';
 73:                                 else
 74:                                         enable <= '0';
 75:                                 end if;
 76:                         elsif ( ready = '1' ) then
 77:                                 enable <= '0';
 78:                         end if;
 79:                 end if;
 80:         end process;
 81:
 82:
 83:         enable_n <= not enable;
 84:
 85:         -- Counter clock enable generator
 86:         -- Produces a positive single-cycle pulse once per microsecond
 87:         timer_clk_gate : cpt_clk_gate
 88:         port map (
 89:                 i_clk => i_clk,
 90:                 i_enable => enable,
 91:                 i_div => MCU_FREQUENCY/2000000, -- 1 us per cycle (n.b. cpt_clk_gate divides by 2 internally)
 92:                 o_clk_pgate => clk_pgate,
 93:                 o_clk_ngate => open
 94:         );
 95:
 96:         -- Timer counter
 97:         cycle_counter : cpt_upcounter
 98:         generic map (
 99:                 INIT => 1
100:         )
```

```
101:         port map (
102:                 i_clk => i_clk,
103:                 i_enable => clk_pgate,
104:                 i_lowest => 0,
105:                 i_highest => max_count,
106:                 i_increment => 1,
107:                 i_clear => enable_n,
108:                 i_preset => '0',
109:                 o_count => open,
110:                 o_carry => ready
111:         );
112:
113:
114: end Behavioral;
115:
```

```
    1:
    2: library ieee;
    3: use ieee.std_logic_1164.all;
    4: use ieee.std_logic_unsigned.all;
    5:
    6: entity i2c_master is
    7:   generic(
    8:     input_clk : integer := 50_000_000; --input clock speed from user logic in hz
    9:     bus_clk   : integer := 400_000);   --speed the i2c bus (scl) will run at in hz
   10:   port(
   11:     clk       : in     std_logic;                    --system clock
   12:     reset_n   : in     std_logic;                    --active low reset
   13:     ena       : in     std_logic;                    --latch in command
   14:     addr      : in     std_logic_vector(6 downto 0); --address of target slave
   15:     rw        : in     std_logic;                    --'0' is write, '1' is read
   16:     data_wr   : in     std_logic_vector(7 downto 0); --data to write to slave
   17:     busy      : out    std_logic;                    --indicates transaction in progress
   18:     data_rd   : out    std_logic_vector(7 downto 0); --data read from slave
   19:     ack_error : buffer std_logic;                    --flag if improper acknowledge from slave
   20:     sda       : inout  std_logic;                    --serial data output of i2c bus
   21:     scl       : inout  std_logic);                   --serial clock output of i2c bus
   22: end i2c_master;
   23:
   24: architecture logic of i2c_master is
   25:   constant divider  :  integer := (input_clk/bus_clk)/4; --number of clocks in 1/4 cycle of scl
   26:   type machine is(ready, start, command, slv_ack1, wr, rd, slv_ack2, mstr_ack, stop); --needed states
   27:   signal state         : machine;                        --state machine
   28:   signal data_clk      : std_logic;                      --data clock for sda
   29:   signal data_clk_prev : std_logic;                      --data clock during previous system clock
   30:   signal scl_clk       : std_logic;                      --constantly running internal scl
   31:   signal scl_ena       : std_logic := '0';               --enables internal scl to output
   32:   signal sda_int       : std_logic := '1';               --internal sda
   33:   signal sda_ena_n     : std_logic;                      --enables internal sda to output
   34:   signal addr_rw       : std_logic_vector(7 downto 0);   --latched in address and read/write
   35:   signal data_tx       : std_logic_vector(7 downto 0);   --latched in data to write to slave
   36:   signal data_rx       : std_logic_vector(7 downto 0);   --data received from slave
   37:   signal bit_cnt       : integer range 0 to 7 := 7;      --tracks bit number in transaction
   38:   signal stretch       : std_logic := '0';               --identifies if slave is stretching scl
   39: begin
   40:
   41:   --generate the timing for the bus clock (scl_clk) and the data clock (data_clk)
   42:   process(clk, reset_n)
   43:     variable count  :  integer range 0 to divider*4;  --timing for clock generation
   44:   begin
   45:     if(reset_n = '0') then                --reset asserted
   46:       stretch <= '0';
   47:       count := 0;
   48:     elsif(clk'event and clk = '1') then
   49:       data_clk_prev <= data_clk;          --store previous value of data clock
   50:       if(count = divider*4-1) then        --end of timing cycle
   51:         count := 0;                       --reset timer
   52:       elsif(stretch = '0') then           --clock stretching from slave not detected
   53:         count := count + 1;               --continue clock generation timing
   54:       end if;
   55:       case count is
   56:         when 0 to divider-1 =>            --first 1/4 cycle of clocking
   57:           scl_clk <= '0';
   58:           data_clk <= '0';
   59:         when divider to divider*2-1 =>    --second 1/4 cycle of clocking
   60:           scl_clk <= '0';
   61:           data_clk <= '1';
   62:         when divider*2 to divider*3-1 =>  --third 1/4 cycle of clocking
   63:           scl_clk <= '1';                 --release scl
   64:           if(scl = '0') then              --detect if slave is stretching clock
   65:             stretch <= '1';
   66:           else
   67:             stretch <= '0';
   68:           end if;
   69:           data_clk <= '1';
   70:         when others =>                    --last 1/4 cycle of clocking
   71:           scl_clk <= '1';
   72:           data_clk <= '0';
   73:       end case;
   74:     end if;
   75:   end process;
   76:
   77:   --state machine and writing to sda during scl low (data_clk rising edge)
   78:   process(clk, reset_n)
   79:   begin
   80:     if(reset_n = '0') then                 --reset asserted
   81:       state <= ready;                      --return to initial state
   82:       busy <= '1';                         --indicate not available
   83:       scl_ena <= '0';                      --sets scl high impedance
   84:       sda_int <= '1';                      --sets sda high impedance
   85:       ack_error <= '0';                    --clear acknowledge error flag
   86:       bit_cnt <= 7;                        --restarts data bit counter
   87:       data_rd <= "00000000";               --clear data read port
   88:     elsif(clk'event and clk = '1') then
   89:       if(data_clk = '1' and data_clk_prev = '0') then  --data clock rising edge
   90:         case state is
   91:           when ready =>                      --idle state
   92:             if(ena = '1') then               --transaction requested
   93:               busy <= '1';                   --flag busy
   94:               addr_rw <= addr & rw;          --collect requested slave address and command
   95:               data_tx <= data_wr;            --collect requested data to write
   96:               state <= start;                --go to start bit
   97:             else                             --remain idle
   98:               busy <= '0';                   --unflag busy
   99:               state <= ready;                --remain idle
  100:             end if;
```

```
101:           when start =>                      --start bit of transaction
102:             busy <= '1';                     --resume busy if continuous mode
103:             sda_int <= addr_rw(bit_cnt);     --set first address bit to bus
104:             state <= command;                --go to command
105:           when command =>                    --address and command byte of transaction
106:             if(bit_cnt = 0) then             --command transmit finished
107:               sda_int <= '1';                --release sda for slave acknowledge
108:               bit_cnt <= 7;                  --reset bit counter for "byte" states
109:               state <= slv_ack1;             --go to slave acknowledge (command)
110:             else                             --next clock cycle of command state
111:               bit_cnt <= bit_cnt - 1;        --keep track of transaction bits
112:               sda_int <= addr_rw(bit_cnt-1); --write address/command bit to bus
113:               state <= command;              --continue with command
114:             end if;
115:           when slv_ack1 =>                   --slave acknowledge bit (command)
116:             if(addr_rw(0) = '0') then        --write command
117:               sda_int <= data_tx(bit_cnt);   --write first bit of data
118:               state <= wr;                   --go to write byte
119:             else                             --read command
120:               sda_int <= '1';                --release sda from incoming data
121:               state <= rd;                   --go to read byte
122:             end if;
123:           when wr =>                         --write byte of transaction
124:             busy <= '1';                     --resume busy if continuous mode
125:             if(bit_cnt = 0) then             --write byte transmit finished
126:               sda_int <= '1';                --release sda for slave acknowledge
127:               bit_cnt <= 7;                  --reset bit counter for "byte" states
128:               state <= slv_ack2;             --go to slave acknowledge (write)
129:             else                             --next clock cycle of write state
130:               bit_cnt <= bit_cnt - 1;        --keep track of transaction bits
131:               sda_int <= data_tx(bit_cnt-1); --write next bit to bus
132:               state <= wr;                   --continue writing
133:             end if;
134:           when rd =>                         --read byte of transaction
135:             busy <= '1';                     --resume busy if continuous mode
136:             if(bit_cnt = 0) then             --read byte receive finished
137:               if(ena = '1' and addr_rw = addr & rw) then  --continuing with another read at same address
138:                 sda_int <= '0';              --acknowledge the byte has been received
139:               else                           --stopping or continuing with a write
140:                 sda_int <= '1';              --send a no-acknowledge (before stop or repeated start)
141:               end if;
142:               bit_cnt <= 7;                  --reset bit counter for "byte" states
143:               data_rd <= data_rx;            --output received data
144:               state <= mstr_ack;             --go to master acknowledge
145:             else                             --next clock cycle of read state
146:               bit_cnt <= bit_cnt - 1;        --keep track of transaction bits
147:               state <= rd;                   --continue reading
148:             end if;
149:           when slv_ack2 =>                   --slave acknowledge bit (write)
150:             if(ena = '1') then               --continue transaction
151:               busy <= '0';                   --continue is accepted
152:               addr_rw <= addr & rw;          --collect requested slave address and command
153:               data_tx <= data_wr;            --collect requested data to write
154:               if(addr_rw = addr & rw) then   --continue transaction with another write
155:                 sda_int <= data_wr(bit_cnt); --write first bit of data
156:                 state <= wr;                 --go to write byte
157:               else                           --continue transaction with a read or new slave
158:                 state <= start;              --go to repeated start
159:               end if;
160:             else                             --complete transaction
161:               state <= stop;                 --go to stop bit
162:             end if;
163:           when mstr_ack =>                   --master acknowledge bit after a read
164:             if(ena = '1') then               --continue transaction
165:               busy <= '0';                   --continue is accepted and data received is available on bus
166:               addr_rw <= addr & rw;          --collect requested slave address and command
167:               data_tx <= data_wr;            --collect requested data to write
168:               if(addr_rw = addr & rw) then   --continue transaction with another read
169:                 sda_int <= '1';              --release sda from incoming data
170:                 state <= rd;                 --go to read byte
171:               else                           --continue transaction with a write or new slave
172:                 state <= start;              --repeated start
173:               end if;
174:             else                             --complete transaction
175:               state <= stop;                 --go to stop bit
176:             end if;
177:           when stop =>                       --stop bit of transaction
178:             busy <= '0';                     --unflag busy
179:             state <= ready;                  --go to idle state
180:         end case;
181:       elsif(data_clk = '0' and data_clk_prev = '1') then  --data clock falling edge
182:         case state is
183:           when start =>
184:             if(scl_ena = '0') then                  --starting new transaction
185:               scl_ena <= '1';                       --enable scl output
186:               ack_error <= '0';                     --reset acknowledge error output
187:             end if;
188:           when slv_ack1 =>                          --receiving slave acknowledge (command)
189:             if(sda /= '0' or ack_error = '1') then  --no-acknowledge or previous no-acknowledge
190:               ack_error <= '1';                     --set error output if no-acknowledge
191:             end if;
192:           when rd =>                                --receiving slave data
193:             data_rx(bit_cnt) <= sda;                --receive current slave data bit
194:           when slv_ack2 =>                          --receiving slave acknowledge (write)
195:             if(sda /= '0' or ack_error = '1') then  --no-acknowledge or previous no-acknowledge
196:               ack_error <= '1';                     --set error output if no-acknowledge
197:             end if;
198:           when stop =>
199:             scl_ena <= '0';                         --disable scl
200:           when others =>
```

```
201:             null;
202:         end case;
203:       end if;
204:     end if;
205:   end process;
206:
207:   --set sda output
208:   with state select
209:     sda_ena_n <= data_clk_prev when start,     --generate start condition
210:                  not data_clk_prev when stop,  --generate stop condition
211:                  sda_int when others;          --set to internal sda signal
212:
213:   --set scl and sda outputs
214:   scl <= '0' when (scl_ena = '1' and scl_clk = '0') else 'z';
215:   sda <= '0' when sda_ena_n = '0' else 'z';
216:
217: end logic;
```

```
  1: --
  2: -- MCU package
  3: --
  4:
  5: library ieee;
  6: use ieee.std_logic_1164.all;
  7:
  8: library mctl;
  9: use mctl.pkg_mctl.all;
 10:
 11: library cctl;
 12: use cctl.pkg_cctl.all;
 13: use cctl.pkg_ovm.all;
 14:
 15: library mcu;
 16:
 17: package pkg_mcu is
 18:         -- ================================================= --
 19:         -- GPIO assignments
 20:         -- ================================================= --
 21:
 22:         -- These must match the #defines in ..SW/src/iobus.h
 23:         constant N_GPIOS : integer := 16#C0#;
 24:
 25:         -- GPIOs between 0x00 and 0x7F are external to cpt_mcu
 26:
 27:         constant GPIO_ERROR_LED                 : integer := 16#00#;
 28:         constant GPIO_LEDS1                             : integer := 16#01#;
 29:         constant GPIO_LEDS2                             : integer := 16#02#;
 30:
 31:         constant GPIO_ERROR_LED_SRC             : integer := 16#10#;
 32:         constant GPIO_LEDS_SRC                  : integer := 16#11#;
 33:         constant GPIO_LEDCLK_DIV                : integer := 16#12#;
 34:         constant GPIO_LED_LATCH_DIV             : integer := 16#13#;
 35:
 36:         constant GPIO_FLASH_CLK_DIV             : integer := 16#18#;
 37:         constant GPIO_FLASH_ON                  : integer := 16#19#;
 38:         constant GPIO_FLASH_MAX                 : integer := 16#1A#;
 39:
 40:         constant GPIO_DEBUG0                    : integer := 16#20#;
 41:         constant GPIO_DEBUG                             : integer := 16#21#;
 42:         constant GPIO_DEBUG_SRC                 : integer := 16#22#;
 43:
 44:         constant GPIO_PROBE_ENABLE              : integer := 16#30#;
 45:         constant GPIO_PROBE_CLEAR               : integer := 16#31#;
 46:         constant GPIO_PROBE_SRC                 : integer := 16#32#;
 47:         constant GPIO_PROBE_LATCH_DIV   : integer := 16#33#;
 48:         constant GPIO_PROBE_LOW                 : integer := 16#34#;
 49:         constant GPIO_PROBE_HIGH                : integer := 16#35#;
 50:         constant GPIO_PROBE_FALL                : integer := 16#36#;
 51:         constant GPIO_PROBE_RISE                : integer := 16#37#;
 52:
 53:
 54:         constant GPIO_OVM_BRAM_ENABLE           : integer := 16#40#;
 55:         constant GPIO_OVM_MUX_ENABLE            : integer := 16#41#;
 56:
 57:         constant GPIO_OVM0_LINE_OFFSET  : integer := 16#44#;
 58:         constant GPIO_OVM1_LINE_OFFSET  : integer := 16#45#;
 59:         constant GPIO_OVM2_LINE_OFFSET  : integer := 16#46#;
 60:         constant GPIO_OVM3_LINE_OFFSET  : integer := 16#47#;
 61:
 62:         constant GPIO_OVM_FRAME_ADDR0   : integer := 16#48#;
 63:         constant GPIO_OVM_FRAME_ADDR1   : integer := 16#49#;
 64:         constant GPIO_OVM_FRAME_ADDR2   : integer := 16#4A#;
 65:         constant GPIO_OVM_FRAME_ADDR3   : integer := 16#4B#;
 66:
 67:         constant GPIO_OVM_PCLK                  : integer := 16#4C#;
 68:         constant GPIO_OVM_HREF                  : integer := 16#4D#;
 69:         constant GPIO_OVM_VSYNC                 : integer := 16#4E#;
 70:
 71:
 72:         constant GPIO_RAM_ERROR_STATUS0 : integer := 16#50#;
 73:         constant GPIO_RAM_ERROR_STATUS1 : integer := 16#51#;
 74:         constant GPIO_RAM_ERROR_STATUS2 : integer := 16#52#;
 75:         constant GPIO_RAM_ERROR_STATUS3 : integer := 16#53#;
 76:         constant GPIO_RAM_STATUS                : integer := 16#54#;
 77:
 78:         constant GPIO_VGA_FIXED_ENABLE  : integer := 16#60#;
 79:         constant GPIO_VGA_SRC                   : integer := 16#61#;
 80:         constant GPIO_VGA_TEST_ENABLE   : integer := 16#62#;
 81:         constant GPIO_VGA_TEST_MODE             : integer := 16#63#;
 82:
 83:         constant GPIO_VGA_FRAME_ADDR0   : integer := 16#64#;
 84:         constant GPIO_VGA_FRAME_ADDR1   : integer := 16#65#;
 85:         constant GPIO_VGA_FRAME_ADDR2   : integer := 16#66#;
 86:         constant GPIO_VGA_FRAME_ADDR3   : integer := 16#67#;
 87:
 88:         constant GPIO_VGA_MID_LINE_OFFSET       : integer := 16#68#;
 89:
 90:
 91:
 92:         -- GPIOs between 0x80 and 0xFF are internal to cpt_mcu
 93:
 94:         constant GPIO_OVM_ENABLE                : integer := 16#80#;
 95:         constant GPIO_OVM_XVCLK_DIV     : integer := 16#81#;
 96:         constant GPIO_OVM_SCL_CLK_DIV   : integer := 16#82#;
 97:         constant GPIO_OVM_DEV_ADDR              : integer := 16#83#;
 98:
 99:         constant GPIO_UART_BAUD_DIV     : integer := 16#90#;
100:         constant GPIO_UART_RX_SRC               : integer := 16#91#;
```

```
101:         constant GPIO_UART_TX_SRC              : integer := 16#92#;
102:         constant GPIO_UART_STATUS              : integer := 16#93#;
103:
104:         constant GPIO_WIFI_BAUD_DIV            : integer := 16#A0#;
105:         constant GPIO_WIFI_RX_SRC              : integer := 16#A1#;
106:         constant GPIO_WIFI_TX_SRC              : integer := 16#A2#;
107:         constant GPIO_WIFI_STATUS              : integer := 16#A3#;
108:
109:         constant GPIO_WIFI_ENABLE              : integer := 16#A4#;
110:
111:         constant GPIO_WIFI_RXD                 : integer := 16#A5#;
112:         constant GPIO_WIFI_RXD_OUTPUT_ENABLE   : integer := 16#A6#;
113:
114:         constant GPIO_WIFI_GPIO                : integer := 16#A7#;
115:         constant GPIO_WIFI_GPIO_OUTPUT_ENABLE  : integer := 16#A8#;
116:
117:         constant GPIO_DEBUG_OUTPUT_ENABLE              : integer := 16#B0#;
118:
119:
120:         -- ================================================= --
121:         -- Type definitions
122:         -- ================================================= --
123:
124:         --subtype typ_mcu_word is std_logic_vector(31 downto 0);
125:         type typ_mcu_word_array is array (0 to N_GPIOS-1) of std_logic_vector(31 downto 0);
126:
127:         -- Microblaze IOBus signals (see ds865 pg3)
128:         -- Master: microblaze
129:         -- Slaves: custom peripherals
130:         type typ_mcu_iobus_miso is record
131:                 read_data : std_logic_vector(31 downto 0);
132:                 ready : std_logic;
133:         end record;
134:
135:         constant init_mcu_iobus_miso : typ_mcu_iobus_miso := (
136:                 read_data => x"FFFFFFFF",
137:                 ready => '0'
138:         );
139:
140:         type typ_mcu_iobus_mosi is record
141:                 addr_strobe : std_logic;
142:                 read_strobe : std_logic;
143:                 write_strobe : std_logic;
144:                 address : std_logic_vector(31 downto 0);
145:                 byte_enable : std_logic_vector(3 downto 0);
146:                 write_data : std_logic_vector(31 downto 0);
147:         end record;
148:
149:         constant init_mcu_iobus_mosi : typ_mcu_iobus_mosi := (
150:                 addr_strobe => '0',
151:                 read_strobe => '0',
152:                 write_strobe => '0',
153:                 address => x"FFFFFFFF",
154:                 byte_enable => x"F",
155:                 write_data => x"FFFFFFFF"
156:         );
157:
158:
159:         -- ================================================= --
160:         -- Component definitions
161:         -- ================================================= --
162:
163:         -- Soft-core processor
164:         -- References cpt_microblaze.xco generated core
165:         component cpt_microblaze
166:                 port (
167:                         Clk : IN STD_LOGIC;
168:                         Reset : IN STD_LOGIC;
169:                         IO_Addr_Strobe : OUT STD_LOGIC;
170:                         IO_Read_Strobe : OUT STD_LOGIC;
171:                         IO_Write_Strobe : OUT STD_LOGIC;
172:                         IO_Address : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
173:                         IO_Byte_Enable : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
174:                         IO_Write_Data : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
175:                         IO_Read_Data : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
176:                         IO_Ready : IN STD_LOGIC;
177:                         UART_Rx : IN STD_LOGIC;
178:                         UART_Tx : OUT STD_LOGIC;
179:                         UART_Interrupt : OUT STD_LOGIC;
180:                         GPO1 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
181:                         GPO2 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
182:                         GPO3 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
183:                         GPO4 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
184:                         GPI1 : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
185:                         GPI1_Interrupt : OUT STD_LOGIC;
186:                         GPI2 : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
187:                         GPI2_Interrupt : OUT STD_LOGIC;
188:                         GPI3 : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
189:                         GPI3_Interrupt : OUT STD_LOGIC;
190:                         GPI4 : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
191:                         GPI4_Interrupt : OUT STD_LOGIC;
192:                         INTC_Interrupt : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
193:                         INTC_IRQ : OUT STD_LOGIC
194:                 );
195:         end component;
196:
197:         -- Data link between the Microblaze IOBus and external RAM via the MCB
198:         component cpt_iobus_mport
199:                 generic (
200:                         DEVICE_ID : std_logic_vector(31 downto 0);
```

```
201:                        DEVICE_ID_MASK : std_logic_vector(31 downto 0)
202:                );
203:                port (
204:                        i_clk : in std_logic;
205:                        i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
206:                        o_mcu_iobus_miso : out typ_mcu_iobus_miso;
207:                        i_mctl_mport_mosi : in typ_mctl_mport_mosi;
208:                        o_mctl_mport_miso : out typ_mctl_mport_miso
209:                );
210:        end component;
211:
212:        component cpt_i2c is
213:                port (
214:                        i_clk : in std_logic;
215:                        i_enable : in std_logic;
216:                        i_scl_clk_div : in integer;
217:                        i_addr : in std_logic_vector(6 downto 0);
218:                        o_rd_data : out std_logic_vector(7 downto 0);
219:                        o_rd_data_strobe : out std_logic;
220:                        i_rd_start : in std_logic;
221:                        o_rd_done : out std_logic;
222:                        i_wr_data_available : in std_logic;
223:                        i_wr_data : in std_logic_vector(7 downto 0);
224:                        o_wr_data_strobe : out std_logic;
225:                        i_wr_start : in std_logic;
226:                        o_wr_done : out std_logic;
227:                        io_i2c_scl : inout std_logic;
228:                        io_i2c_sda : inout std_logic
229:                );
230:        end component;
231:
232:        component cpt_iobus_sccb is
233:                generic (
234:                        DEVICE_ID : std_logic_vector(31 downto 0);
235:                        DEVICE_ID_MASK : std_logic_vector(31 downto 0)
236:                );
237:                port (
238:                        i_clk : in std_logic;
239:                        i_enable : in std_logic;
240:                        i_iobus_mosi : in typ_mcu_iobus_mosi;
241:                        o_iobus_miso : out typ_mcu_iobus_miso;
242:                        i_dev_addr : in std_logic_vector(6 downto 0);
243:                        i_scl_clk_div : in integer;
244:                        io_scl : inout std_logic;
245:                        io_sda : inout std_logic
246:                );
247:        end component;
248:
249:        -- Microblaze core with custom peripherals
250:        component cpt_mcu is
251:                generic (
252:                        INCLUDE_IOBUS_MPORT : string := "TRUE";
253:                        INCLUDE_SCCB : string := "TRUE"
254:                );
255:                port (
256:                        i_clk : in std_logic;
257:                        i_reset : in std_logic;
258:
259:                        o_debug_output_enable : out std_logic;
260:                        o_debug_src : out integer range 0 to 15;
261:
262:                        i_mctl_mport_mosi : in typ_mctl_mport_mosi;
263:                        o_mctl_mport_miso : out typ_mctl_mport_miso;
264:
265:                        io_ovm0_sccb_bidir : inout typ_ovm_sccb_bidir;
266:                        o_ovm0_sccb_mosi : out typ_ovm_sccb_mosi;
267:                        io_ovm1_sccb_bidir : inout typ_ovm_sccb_bidir;
268:                        o_ovm1_sccb_mosi : out typ_ovm_sccb_mosi;
269:                        io_ovm2_sccb_bidir : inout typ_ovm_sccb_bidir;
270:                        o_ovm2_sccb_mosi : out typ_ovm_sccb_mosi;
271:                        io_ovm3_sccb_bidir : inout typ_ovm_sccb_bidir;
272:                        o_ovm3_sccb_mosi : out typ_ovm_sccb_mosi;
273:
274:                        i_uart_rx : in std_logic := '1';
275:                        o_uart_tx : out std_logic := '1';
276:
277:                        o_wifi_txd : inout std_logic;
278:                        io_wifi_rxd : inout std_logic;
279:                        o_wifi_rst : out std_logic;
280:                        io_wifi_gpio0 : inout std_logic;
281:                        io_wifi_gpio2 : inout std_logic;
282:                        o_wifi_ch_pd : out std_logic;
283:
284: --                     i_gp1i : in std_logic_vector(31 downto 0);
285: --                     i_gp2i : in std_logic_vector(31 downto 0);
286: --                     i_gp3i : in std_logic_vector(31 downto 0);
287: --                     i_gp4i : in std_logic_vector(31 downto 0);
288: --                     o_gp1o : out std_logic_vector(31 downto 0);
289: --                     o_gp2o : out std_logic_vector(31 downto 0);
290: --                     o_gp3o : out std_logic_vector(31 downto 0);
291: --                     o_gp4o : out std_logic_vector(31 downto 0);
292:
293:                        i_gpi : in typ_mcu_word_array;
294:                        o_gpo : out typ_mcu_word_array;
295:
296:                        i_intc_interrupt : in std_logic_vector(7 downto 0);
297:                        o_intc_irq : out std_logic_vector(31 downto 0)
298:                );
299:        end component;
300:
```

```
301:         component cpt_timer is
302:                 generic (
303:                         DEVICE_ID : std_logic_vector(31 downto 0);
304:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0);
305:                         MCU_FREQUENCY : integer
306:                 );
307:                 port (
308:                         i_clk : in std_logic;
309:                         i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
310:                         o_mcu_iobus_miso : out typ_mcu_iobus_miso
311:                 );
312:         end component;
313:
314:         component cpt_counter is
315:                 generic (
316:                         DEVICE_ID : std_logic_vector(31 downto 0);
317:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0);
318:                         MCU_FREQUENCY : integer
319:                 );
320:                 port (
321:                         i_clk : in std_logic;
322:                         i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
323:                         o_mcu_iobus_miso : out typ_mcu_iobus_miso
324:                 );
325:         end component;
326:
327:         component cpt_gpio is
328:                 generic (
329:                         DEVICE_ID : std_logic_vector(31 downto 0);
330:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0);
331:                         N_GPIOS : integer
332:                 );
333:                 port (
334:                         i_clk : in std_logic;
335:                         i_reset : in std_logic;
336:                         i_mcu_iobus_mosi : in typ_mcu_iobus_mosi;
337:                         o_mcu_iobus_miso : out typ_mcu_iobus_miso;
338:                         i_gpi : in typ_mcu_word_array;
339:                         o_gpo : out typ_mcu_word_array
340:                 );
341:         end component;
342:
343:         component cpt_sccb_master is
344:                 generic (
345:                         DEVICE_ID : std_logic_vector(31 downto 0);
346:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0)
347:                 );
348:                 port (
349:                         i_clk : in std_logic;
350:                         i_enable : in std_logic;
351:                         i_iobus_mosi : in typ_mcu_iobus_mosi;
352:                         o_iobus_miso : out typ_mcu_iobus_miso;
353:                         i_scl_div : in integer;
354:                         io_sccb_bidir : inout typ_ovm_sccb_bidir;
355:                         o_sccb_mosi : out typ_ovm_sccb_mosi
356:                 );
357:         end component;
358: end pkg_mcu;
359:
360: package body pkg_mcu is
361:
362: end pkg_mcu;
```

```
--
-- Memory controller package
-- Reference ug388_Spartan6_MemoryControlBlock.pdf
--

library ieee;
use ieee.std_logic_1164.all;

library work;

package pkg_testing is
	constant C3_NUM_DQ_PINS         : integer := 16;        -- External memory data width.
	constant C3_MEM_ADDR_WIDTH      : integer := 13;        -- External memory address width.
	constant C3_MEM_BANKADDR_WIDTH  : integer := 2;         -- External memory bank address width.
	constant C3_MEM_ADDR_ORDER      : string  := "ROW_BANK_COLUMN";
	constant C3_P0_MASK_SIZE        : integer := 4;
	constant C3_P0_DATA_PORT_SIZE   : integer := 32;
	constant C3_P1_MASK_SIZE        : integer := 4;
	constant C3_P1_DATA_PORT_SIZE   : integer := 32;




	component cpt_upcounter_testing is
		generic (
			INIT : integer := -1
		);
		port (
			i_clk : in std_logic;
			i_enable : in std_logic;
			i_lowest : in integer;
			i_highest : in integer;
			i_increment : in integer;
			i_clear : in std_logic;
			i_preset : in std_logic;
			o_count : out integer := INIT;
			o_carry : out std_logic
		);
	end component;


	-- Camera video

	type typ_ovm_sccb_bidir is record
		scl : std_logic;
		sda : std_logic;
	end record;

	constant init_ovm_sccb_bidir : typ_ovm_sccb_bidir := (
		scl => 'Z',
		sda => 'Z'
	);


	type typ_ovm_sccb_mosi is record
		pwdn : std_logic;
		xvclk : std_logic;
	end record;

	constant init_ovm_sccb_mosi : typ_ovm_sccb_mosi := (
		pwdn => '1',
		xvclk => '0'
	);


	type typ_ovm_video_miso is record
		pclk : std_logic;
		data : std_logic_vector(7 downto 0);
		href : std_logic;
		vsync : std_logic;
	end record;

	constant init_ovm_video_miso : typ_ovm_video_miso := (
		pclk => '0',
		data => (others => '0'),
		href => '0',
		vsync => '0'
	);



	-- MCB port

	type typ_mctl_mport_cmd_miso is record
		clk : std_logic;
		en : std_logic;
		instr : std_logic_vector(2 downto 0);
		bl : std_logic_vector(5 downto 0);
		byte_addr : std_logic_vector(29 downto 0);
	end record;

	constant init_mctl_mport_cmd_miso : typ_mctl_mport_cmd_miso := (
		clk => '0',
		en => '0',
		instr => (others => '0'),
		bl => (others => '0'),
		byte_addr => (others => '0')
	);

```

```
101:
102:         type typ_mctl_mport_wr_miso is record
103:                 clk : std_logic;
104:                 en : std_logic;
105:                 mask : std_logic_vector(3 downto 0);
106:                 data : std_logic_vector(31 downto 0);
107:         end record;
108:
109:         constant init_mctl_mport_wr_miso : typ_mctl_mport_wr_miso := (
110:                 clk => '0',
111:                 en => '0',
112:                 mask => (others => '0'),
113:                 data => (others => '0')
114:         );
115:
116:
117:         type typ_mctl_mport_rd_miso is record
118:                 clk : std_logic;
119:                 en : std_logic;
120:         end record;
121:
122:         constant init_mctl_mport_rd_miso : typ_mctl_mport_rd_miso := (
123:                 clk => '0',
124:                 en => '0'
125:         );
126:
127:
128:         type typ_mctl_mport_miso is record
129:                 cmd : typ_mctl_mport_cmd_miso;
130:                 wr : typ_mctl_mport_wr_miso;
131:                 rd : typ_mctl_mport_rd_miso;
132:         end record;
133:
134:         constant init_mctl_mport_miso : typ_mctl_mport_miso := (
135:                 cmd => init_mctl_mport_cmd_miso,
136:                 wr => init_mctl_mport_wr_miso,
137:                 rd => init_mctl_mport_rd_miso
138:         );
139:
140:
141:         type typ_mctl_mport_cmd_mosi is record
142:                 empty : std_logic;
143:                 full : std_logic;
144:         end record;
145:
146:         constant init_mctl_mport_cmd_mosi : typ_mctl_mport_cmd_mosi := (
147:                 empty => '0',
148:                 full => '0'
149:         );
150:
151:
152:         type typ_mctl_mport_wr_mosi is record
153:                 empty : std_logic;
154:                 full : std_logic;
155:                 count : std_logic_vector(6 downto 0);
156:                 underrun : std_logic;
157:                 error : std_logic;
158:         end record;
159:
160:         constant init_mctl_mport_wr_mosi : typ_mctl_mport_wr_mosi := (
161:                 empty => '0',
162:                 full => '0',
163:                 count => (others => '0'),
164:                 underrun => '0',
165:                 error => '0'
166:         );
167:
168:
169:         type typ_mctl_mport_rd_mosi is record
170:                 data : std_logic_vector(31 downto 0);
171:                 empty : std_logic;
172:                 full : std_logic;
173:                 count : std_logic_vector(6 downto 0);
174:                 overflow : std_logic;
175:                 error : std_logic;
176:         end record;
177:
178:         constant init_mctl_mport_rd_mosi : typ_mctl_mport_rd_mosi := (
179:                 data => (others => '0'),
180:                 empty => '0',
181:                 full => '0',
182:                 count => (others => '0'),
183:                 overflow => '0',
184:                 error => '0'
185:         );
186:
187:
188:         type typ_mctl_mport_mosi is record
189:                 cmd : typ_mctl_mport_cmd_mosi;
190:                 wr : typ_mctl_mport_wr_mosi;
191:                 rd : typ_mctl_mport_rd_mosi;
192:         end record;
193:
194:         constant init_mctl_mport_mosi : typ_mctl_mport_mosi := (
195:                 cmd => init_mctl_mport_cmd_mosi,
196:                 wr => init_mctl_mport_wr_mosi,
197:                 rd => init_mctl_mport_rd_mosi
198:         );
199:
200:
```

```
  201:         type typ_mctl_ram_bidir is record
  202:                 dq : std_logic_vector(C3_NUM_DQ_PINS-1 downto 0);
  203:                 udqs : std_logic;
  204:                 dqs : std_logic;
  205:         end record;
  206:
  207:         type typ_mctl_ram_mosi is record
  208:                 a : std_logic_vector(C3_MEM_ADDR_WIDTH-1 downto 0);
  209:                 ba : std_logic_vector(C3_MEM_BANKADDR_WIDTH-1 downto 0);
  210:                 cke : std_logic;
  211:                 ras_n : std_logic;
  212:                 cas_n : std_logic;
  213:                 we_n : std_logic;
  214:                 dm : std_logic;
  215:                 udm : std_logic;
  216:                 ck : std_logic;
  217:                 ck_n : std_logic;
  218:         end record;
  219:
  220:         component cpt_mctl_wrapper is
  221:                 generic (
  222:                         INCLUDE_MCTL_CHIPSCOPE  : string  := "TRUE";
  223:                         C3_MEMCLK_PERIOD        : integer := 6000;
  224:                                                                                 -- Memory data transfer clock period.
  225:                         C3_CALIB_SOFT_IP        : string := "TRUE";
  226:                                                                                 -- # = TRUE, Enables the soft calibration logic,
  227:                                                                                 -- # = FALSE, Disables the soft calibration logic.
  228:                         C3_SIMULATION           : string := "FALSE"
  229:                                                                                 -- # = TRUE, Simulating the design. Useful to reduce
the simulation time,
  230:                                                                                 -- # = FALSE, Implementing the design.
  231:                         --DEBUG_EN                : integer := 1
  232:                                                                                 -- # = 1, Enable debug signals/controls,
  233:                                                                                 --   = 0, Disable debug signals/controls.
  234:                 );
  235:                 port (
  236:                         mcb3_rzq                : inout std_logic;
  237:
  238:                         c3_sys_clk              : in std_logic;
  239:                         c3_sys_rst_i            : in std_logic;
  240:                         c3_calib_done           : out std_logic;
  241:                         c3_clk0                 : out std_logic;
  242:                         c3_rst0                 : out std_logic;
  243:
  244:                         clk_108                 : out std_logic;
  245:                         clk_108_n               : out std_logic;
  246:
  247:                         ram_bidir   : inout typ_mctl_ram_bidir;
  248:                         ram_mosi    : out typ_mctl_ram_mosi;
  249:
  250:                         mport0_miso : in typ_mctl_mport_miso;
  251:                         mport0_mosi : out typ_mctl_mport_mosi;
  252:
  253:                         mport1_miso : in typ_mctl_mport_miso;
  254:                         mport1_mosi : out typ_mctl_mport_mosi;
  255:
  256:                         mport2_miso : in typ_mctl_mport_miso;
  257:                         mport2_mosi : out typ_mctl_mport_mosi;
  258:
  259:                         mport3_miso : in typ_mctl_mport_miso;
  260:                         mport3_mosi : out typ_mctl_mport_mosi
  261:                 );
  262:         end component;
  263:
  264:         component cpt_mctl is
  265:                 generic (
  266:                         INCLUDE_MCTL_CHIPSCOPE  : string;
  267:                         C3_P0_MASK_SIZE         : integer;
  268:                         C3_P0_DATA_PORT_SIZE    : integer;
  269:                         C3_P1_MASK_SIZE         : integer;
  270:                         C3_P1_DATA_PORT_SIZE    : integer;
  271:
  272:                         C3_MEMCLK_PERIOD        : integer;
  273:                         C3_RST_ACT_LOW          : integer;
  274:                         C3_INPUT_CLK_TYPE       : string;
  275:                         DEBUG_EN                : integer;
  276:
  277:                         C3_CALIB_SOFT_IP        : string;
  278:                         C3_SIMULATION           : string;
  279:                         C3_MEM_ADDR_ORDER       : string;
  280:                         C3_NUM_DQ_PINS          : integer;
  281:                         C3_MEM_ADDR_WIDTH       : integer;
  282:                         C3_MEM_BANKADDR_WIDTH   : integer
  283:                 );
  284:                 port (
  285:                         mcb3_dram_dq            : inout std_logic_vector(C3_NUM_DQ_PINS-1 downto 0);
  286:                         mcb3_dram_dqs           : inout std_logic;
  287:                         mcb3_dram_udqs          : inout std_logic;
  288:                         mcb3_dram_a             : out std_logic_vector(C3_MEM_ADDR_WIDTH-1 downto 0);
  289:                         mcb3_dram_ba            : out std_logic_vector(C3_MEM_BANKADDR_WIDTH-1 downto 0);
  290:                         mcb3_dram_ras_n         : out std_logic;
  291:                         mcb3_dram_cas_n         : out std_logic;
  292:                         mcb3_dram_we_n          : out std_logic;
  293:                         mcb3_dram_cke           : out std_logic;
  294:                         mcb3_dram_dm            : out std_logic;
  295:                         mcb3_dram_ck            : out std_logic;
  296:                         mcb3_dram_udm           : out std_logic;
  297:                         mcb3_dram_ck_n          : out std_logic;
  298:
  299:                         mcb3_rzq                : inout std_logic;
```

```
300:                    c3_sys_clk             : in std_logic;
301:                    c3_sys_rst_i           : in std_logic;
302:                    c3_calib_done          : out std_logic;
303:                    c3_clk0                : out std_logic;
304:                    c3_rst0                : out std_logic;
305:
306:                    clk_108                : out std_logic;
307:                    clk_108_n              : out std_logic;
308:
309:                    c3_p0_cmd_clk          : in std_logic;
310:                    c3_p0_cmd_en           : in std_logic;
311:                    c3_p0_cmd_instr        : in std_logic_vector(2 downto 0);
312:                    c3_p0_cmd_bl           : in std_logic_vector(5 downto 0);
313:                    c3_p0_cmd_byte_addr    : in std_logic_vector(29 downto 0);
314:                    c3_p0_cmd_empty        : out std_logic;
315:                    c3_p0_cmd_full         : out std_logic;
316:                    c3_p0_wr_clk           : in std_logic;
317:                    c3_p0_wr_en            : in std_logic;
318:                    c3_p0_wr_mask          : in std_logic_vector(C3_P0_MASK_SIZE - 1 downto 0);
319:                    c3_p0_wr_data          : in std_logic_vector(C3_P0_DATA_PORT_SIZE - 1 downto 0);
320:                    c3_p0_wr_full          : out std_logic;
321:                    c3_p0_wr_empty         : out std_logic;
322:                    c3_p0_wr_count         : out std_logic_vector(6 downto 0);
323:                    c3_p0_wr_underrun      : out std_logic;
324:                    c3_p0_wr_error         : out std_logic;
325:                    c3_p0_rd_clk           : in std_logic;
326:                    c3_p0_rd_en            : in std_logic;
327:                    c3_p0_rd_data          : out std_logic_vector(C3_P0_DATA_PORT_SIZE - 1 downto 0);
328:                    c3_p0_rd_full          : out std_logic;
329:                    c3_p0_rd_empty         : out std_logic;
330:                    c3_p0_rd_count         : out std_logic_vector(6 downto 0);
331:                    c3_p0_rd_overflow      : out std_logic;
332:                    c3_p0_rd_error         : out std_logic;
333:                    c3_p1_cmd_clk          : in std_logic;
334:                    c3_p1_cmd_en           : in std_logic;
335:                    c3_p1_cmd_instr        : in std_logic_vector(2 downto 0);
336:                    c3_p1_cmd_bl           : in std_logic_vector(5 downto 0);
337:                    c3_p1_cmd_byte_addr    : in std_logic_vector(29 downto 0);
338:                    c3_p1_cmd_empty        : out std_logic;
339:                    c3_p1_cmd_full         : out std_logic;
340:                    c3_p1_wr_clk           : in std_logic;
341:                    c3_p1_wr_en            : in std_logic;
342:                    c3_p1_wr_mask          : in std_logic_vector(C3_P1_MASK_SIZE - 1 downto 0);
343:                    c3_p1_wr_data          : in std_logic_vector(C3_P1_DATA_PORT_SIZE - 1 downto 0);
344:                    c3_p1_wr_full          : out std_logic;
345:                    c3_p1_wr_empty         : out std_logic;
346:                    c3_p1_wr_count         : out std_logic_vector(6 downto 0);
347:                    c3_p1_wr_underrun      : out std_logic;
348:                    c3_p1_wr_error         : out std_logic;
349:                    c3_p1_rd_clk           : in std_logic;
350:                    c3_p1_rd_en            : in std_logic;
351:                    c3_p1_rd_data          : out std_logic_vector(C3_P1_DATA_PORT_SIZE - 1 downto 0);
352:                    c3_p1_rd_full          : out std_logic;
353:                    c3_p1_rd_empty         : out std_logic;
354:                    c3_p1_rd_count         : out std_logic_vector(6 downto 0);
355:                    c3_p1_rd_overflow      : out std_logic;
356:                    c3_p1_rd_error         : out std_logic;
357:                    c3_p2_cmd_clk          : in std_logic;
358:                    c3_p2_cmd_en           : in std_logic;
359:                    c3_p2_cmd_instr        : in std_logic_vector(2 downto 0);
360:                    c3_p2_cmd_bl           : in std_logic_vector(5 downto 0);
361:                    c3_p2_cmd_byte_addr    : in std_logic_vector(29 downto 0);
362:                    c3_p2_cmd_empty        : out std_logic;
363:                    c3_p2_cmd_full         : out std_logic;
364:                    c3_p2_wr_clk           : in std_logic;
365:                    c3_p2_wr_en            : in std_logic;
366:                    c3_p2_wr_mask          : in std_logic_vector(3 downto 0);
367:                    c3_p2_wr_data          : in std_logic_vector(31 downto 0);
368:                    c3_p2_wr_full          : out std_logic;
369:                    c3_p2_wr_empty         : out std_logic;
370:                    c3_p2_wr_count         : out std_logic_vector(6 downto 0);
371:                    c3_p2_wr_underrun      : out std_logic;
372:                    c3_p2_wr_error         : out std_logic;
373:                    c3_p2_rd_clk           : in std_logic;
374:                    c3_p2_rd_en            : in std_logic;
375:                    c3_p2_rd_data          : out std_logic_vector(31 downto 0);
376:                    c3_p2_rd_full          : out std_logic;
377:                    c3_p2_rd_empty         : out std_logic;
378:                    c3_p2_rd_count         : out std_logic_vector(6 downto 0);
379:                    c3_p2_rd_overflow      : out std_logic;
380:                    c3_p2_rd_error         : out std_logic;
381:                    c3_p3_cmd_clk          : in std_logic;
382:                    c3_p3_cmd_en           : in std_logic;
383:                    c3_p3_cmd_instr        : in std_logic_vector(2 downto 0);
384:                    c3_p3_cmd_bl           : in std_logic_vector(5 downto 0);
385:                    c3_p3_cmd_byte_addr    : in std_logic_vector(29 downto 0);
386:                    c3_p3_cmd_empty        : out std_logic;
387:                    c3_p3_cmd_full         : out std_logic;
388:                    c3_p3_wr_clk           : in std_logic;
389:                    c3_p3_wr_en            : in std_logic;
390:                    c3_p3_wr_mask          : in std_logic_vector(3 downto 0);
391:                    c3_p3_wr_data          : in std_logic_vector(31 downto 0);
392:                    c3_p3_wr_full          : out std_logic;
393:                    c3_p3_wr_empty         : out std_logic;
394:                    c3_p3_wr_count         : out std_logic_vector(6 downto 0);
395:                    c3_p3_wr_underrun      : out std_logic;
396:                    c3_p3_wr_error         : out std_logic;
397:                    c3_p3_rd_clk           : in std_logic;
398:                    c3_p3_rd_en            : in std_logic;
399:                    c3_p3_rd_data          : out std_logic_vector(31 downto 0);
```

```
400:                        c3_p3_rd_full           : out std_logic;
401:                        c3_p3_rd_empty          : out std_logic;
402:                        c3_p3_rd_count          : out std_logic_vector(6 downto 0);
403:                        c3_p3_rd_overflow       : out std_logic;
404:                        c3_p3_rd_error          : out std_logic
405:                );
406:        end component;
407: end pkg_testing;
408:
409: package body pkg_testing is
410:
411: end pkg_testing;
```

```
  1:
  2: -- sim_ovm.vhd
  3: --
  4: -- Behavioural simulation of OVM7690 CameraCube.
  5: --
  6: -- Implemented:
  7: --              PLL (6MHz xvclk in, 24MHz pclk out)
  8: --              Internal registers
  9: --      Tristate ctrl/video bus
 10: --
 11: -- To be implemented:
 12: --      SCCB register read/write
 13: --              Video readout (RGB/YUV)
 14: --
 15: -- Won't be implemented
 16: --              Image processing
 17: --
 18:
 19:
 20: library ieee;
 21: use ieee.std_logic_1164.all;
 22: --use ieee.std_logic_arith.all;
 23: use ieee.numeric_std.all;
 24:
 25: library unisim;
 26: use unisim.vcomponents.all;
 27:
 28: library work;
 29: use work.pkg_testing.all;
 30:
 31:
 32:
 33: entity sim_ovm_testing is
 34:         generic (
 35:                 SMALL_FRAME : string := "FALSE" -- Image size is reduced by a factor of ~100
 36:         );
 37:         port (
 38:                 i_ovm_sccb_mosi : in typ_ovm_sccb_mosi;
 39:                 io_ovm_sccb_bidir : inout typ_ovm_sccb_bidir;
 40:                 o_ovm_video_miso : out typ_ovm_video_miso
 41:         );
 42: end sim_ovm_testing;
 43:
 44:
 45: architecture Behavioral of sim_ovm_testing is
 46:
 47:         signal xvclk : std_logic := '0';
 48:
 49:         -- PLL
 50:         signal pll_clkfb : std_logic := '0';
 51:         signal pll_reset : std_logic := '1';
 52:         signal pll_lock : std_logic := '0';
 53:         signal pll_lock_n : std_logic := '1';
 54:
 55:         -- System
 56:         signal clk_8M : std_logic := '0';
 57:         signal clk_12M : std_logic := '0';
 58:         signal clk_24M : std_logic := '0';
 59:         signal clk_48M : std_logic := '0';
 60:         signal reset : std_logic := '1';
 61:
 62:
 63:         -- Register bank (refer to OV7690_CSP3 datasheet)
 64:         type typ_ovm_registers is array (0 to 255) of std_logic_vector(7 downto 0);
 65:
 66:         -- Register addresses
 67:         constant OVM_PIDH       : integer := 16#0A#; -- product ID MSB
 68:         constant OVM_PIDL       : integer := 16#0B#; -- product ID LSB
 69:         constant OVM_REG0C      : integer := 16#0C#; -- vflip,hmirror,BRswap,YUYVswap,busorder,tristate,overlay
 70:         constant OVM_REG0D      : integer := 16#0D#; -- VSstart,VSwidth
 71:         constant OVM_REG0E      : integer := 16#0E#; -- Sleep,Range,Drive
 72:         constant OVM_CLKRC      : integer := 16#11#; -- ExtClk,PreScale
 73:         constant OVM_REG12      : integer := 16#12#; -- Reset,Subsmp,ITU565,RAW,RGBfmt,OUTfmt
 74:         constant OVM_REG16      : integer := 16#16#; -- HsizeLSb,Voff,Hoff
 75:         constant OVM_HSIZE      : integer := 16#18#; -- HsizeMSB
 76:         constant OVM_VSIZE      : integer := 16#1A#; -- Vsize
 77:         constant OVM_MIDH       : integer := 16#1C#; -- mfr. ID MSB
 78:         constant OVM_MIDL       : integer := 16#1D#; -- mfr. ID LSB
 79:         constant OVM_REG28      : integer := 16#28#; -- DATAneg,HRtoHS,HSrev,HRrev,VSedge,VSneg
 80:         constant OVM_PLL                : integer := 16#29#; -- PLLdiv,PLLctl,PLLreset,YAVGsrc
 81:         constant OVM_REG3E      : integer := 16#3E#; -- PCLKgate,PCLKmult
 82:         constant OVM_REG3F      : integer := 16#3F#; -- PCLKrev
 83:         constant OVM_PWC0       : integer := 16#49#; -- DOVDD
 84:         constant OVM_REG62      : integer := 16#62#; -- TESTen,TESTmode
 85:
 86:
 87:         signal ovm_reg : typ_ovm_registers := (
 88:                 OVM_PIDH        => x"76",
 89:                 OVM_PIDL        => x"90",
 90:                 OVM_REG0C       => x"00",
 91:                 OVM_REG0D       => x"44",
 92:                 OVM_REG0E       => x"00",
 93:                 OVM_CLKRC       => x"00",
 94:                 OVM_REG12       => x"11",
 95:                 OVM_REG16       => x"08",
 96:                 OVM_HSIZE       => x"A0",
 97:                 OVM_VSIZE       => x"F0",
 98:                 OVM_MIDH        => x"7F",
 99:                 OVM_MIDL        => x"A2",
100:                 OVM_REG28       => x"00",
```

```
101:                OVM_PLL                 => x"A2",
102:                OVM_REG3E       => x"20",
103:                OVM_REG3F       => x"44",
104:                OVM_PWC0                => x"0D",
105:                OVM_REG62       => x"00",
106:                others => x"00"
107:        );
108:
109:
110:        -- Video generation
111:        signal red_pixel : integer := 0;
112:        signal green_pixel : integer := 0;
113:        signal blue_pixel : integer := 0;
114:
115:
116:
117:        function f(v1:integer; v2:integer) return integer is
118:                begin
119:                if (SMALL_FRAME = "FALSE") then
120:                  return v1;
121:                else
122:                  return v2;
123:                end if;
124:    end function;
125:
126:
127:
128:
129:
130:        -- ----------------------------------
131:        -- Characteristic timing constants
132:        -- ----------------------------------
133:
134:        constant H_ACTIVE_WIDTH : integer := f(640, 12);
135:        constant H_FRONTPORCH_WIDTH : integer := f(54, 2);
136:        constant H_SYNC_WIDTH : integer := f(16, 1);
137:        constant H_BACKPORCH_WIDTH : integer := f(70, 2);
138:
139:        constant V_ACTIVE_WIDTH : integer := f(480, 12);
140:        constant V_FRONTPORCH_WIDTH : integer := f(12, 1);
141:        constant V_SYNC_WIDTH : integer := f(4, 1);
142:        constant V_BACKPORCH_WIDTH : integer := f(16, 2);
143:
144:
145:        -- ------------------------------------------------------
146:        -- Derived timing constants (do not modify without reason)
147:        -- ------------------------------------------------------
148:
149:        constant H_ACTIVE_FIRST : integer := 0;
150:        constant H_ACTIVE_LAST : integer := H_ACTIVE_FIRST + H_ACTIVE_WIDTH - 1;
151:
152:        constant H_FRONTPORCH_FIRST : integer := H_ACTIVE_LAST + 1;
153:        constant H_FRONTPORCH_LAST : integer := H_FRONTPORCH_FIRST + H_FRONTPORCH_WIDTH - 1;
154:
155:        constant H_SYNC_FIRST : integer := H_FRONTPORCH_LAST + 1;
156:        constant H_SYNC_LAST : integer := H_SYNC_FIRST + H_SYNC_WIDTH - 1;
157:
158:        constant H_BACKPORCH_FIRST : integer := H_SYNC_LAST + 1;
159:        constant H_BACKPORCH_LAST : integer := H_BACKPORCH_FIRST + H_BACKPORCH_WIDTH - 1;
160:
161:        constant H_BLANK_FIRST : integer := H_FRONTPORCH_FIRST;
162:        constant H_BLANK_LAST : integer := H_BACKPORCH_LAST;
163:
164:        constant H_FRAME_FIRST : integer := H_ACTIVE_FIRST;
165:        constant H_FRAME_LAST : integer := H_BACKPORCH_LAST;
166:
167:        constant V_ACTIVE_FIRST : integer := 0;
168:        constant V_ACTIVE_LAST : integer := V_ACTIVE_FIRST + V_ACTIVE_WIDTH - 1;
169:
170:        constant V_FRONTPORCH_FIRST : integer := V_ACTIVE_LAST + 1;
171:        constant V_FRONTPORCH_LAST : integer := V_FRONTPORCH_FIRST + V_FRONTPORCH_WIDTH - 1;
172:
173:        constant V_SYNC_FIRST : integer := V_FRONTPORCH_LAST + 1;
174:        constant V_SYNC_LAST : integer := V_SYNC_FIRST + V_SYNC_WIDTH - 1;
175:
176:        constant V_BACKPORCH_FIRST : integer := V_SYNC_LAST + 1;
177:        constant V_BACKPORCH_LAST : integer := V_BACKPORCH_FIRST + V_BACKPORCH_WIDTH - 1;
178:
179:        constant V_BLANK_FIRST : integer := V_FRONTPORCH_FIRST;
180:        constant V_BLANK_LAST : integer := V_BACKPORCH_LAST;
181:
182:        constant V_FRAME_FIRST : integer := V_ACTIVE_FIRST;
183:        constant V_FRAME_LAST : integer := V_BACKPORCH_LAST;
184:
185:
186:        -- Video timing generation
187:        signal h_count : integer := 0;
188:        signal h_active : std_logic := '1';
189:        signal h_sync : std_logic := '0';
190:
191:        signal v_counter_enable : std_logic := '1';
192:        signal v_count : integer := 0;
193:        signal v_active : std_logic := '1';
194:        signal v_sync : std_logic := '0';
195:
196:        signal frame_counter_enable : std_logic := '1';
197:        signal frame_count : integer := 0;
198:        signal frame_active : std_logic := '1';
199:
200:
```

```
201:
202:
203:         signal subpixel : integer := 0;
204:         constant RGB : string := "RGBg";
205:         constant YCrCb : string := "YRyB";
206:
207:
208:
209:         -- Video output
210:         signal data : std_logic_vector(7 downto 0) := (others => '0');
211:         --signal pclk : std_logic := '0';
212:         --signal vsync : std_logic := '0';
213:         --signal href : std_logic := '0';
214:         signal tristate_ctrl : std_logic := '1';
215:         signal tristate_data : std_logic := '1';
216:
217:
218: begin
219:
220:         xvclk <= i_ovm_sccb_mosi.xvclk;
221:
222: --      xvclk_ibufg : IBUFG
223: --      port map (
224: --                      I => i_ovm_sccb_mosi.xvclk,
225: --                      O => xvclk
226: --      );
227:
228:
229:
230:
231:         -- 6 MHz ->
232:         --                      48 MHz
233:         --                      24 MHz -- pixel clock
234:         --                      12 MHz
235:         --                      8 MHz
236:
237:         pll_reset <= ovm_reg(OVM_PLL)(3);
238:
239:    ovm_pll_base : PLL_BASE
240:    generic map (
241:       BANDWIDTH => "OPTIMIZED", -- "HIGH", "LOW" or "OPTIMIZED"
242:       CLKFBOUT_MULT => 16, -- Multiply value for all CLKOUT clock outputs (1-64)
243:       CLKFBOUT_PHASE => 0.0, -- Phase offset in degrees of the clock feedback output (0.0-360.0).
244:       CLKIN_PERIOD => 166.667, -- 6 MHz     -- Input clock period in ns
245:       -- CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for CLKOUT# clock output (1-128)
246:       CLKOUT0_DIVIDE => 2,
247:       CLKOUT1_DIVIDE => 4,
248:       CLKOUT2_DIVIDE => 8,
249:       CLKOUT3_DIVIDE => 12,
250:       CLKOUT4_DIVIDE => 1,
251:       CLKOUT5_DIVIDE => 1,
252:       -- CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for CLKOUT# clock output (0.01-0.99).
253:       CLKOUT0_DUTY_CYCLE => 0.5,
254:       CLKOUT1_DUTY_CYCLE => 0.5,
255:       CLKOUT2_DUTY_CYCLE => 0.5,
256:       CLKOUT3_DUTY_CYCLE => 0.5,
257:       CLKOUT4_DUTY_CYCLE => 0.5,
258:       CLKOUT5_DUTY_CYCLE => 0.5,
259:       -- CLKOUT0_PHASE - CLKOUT5_PHASE: Output phase relationship for CLKOUT# clock output (-360.0-360.0).
260:       CLKOUT0_PHASE => 0.0,
261:       CLKOUT1_PHASE => 0.0,
262:       CLKOUT2_PHASE => 0.0,
263:       CLKOUT3_PHASE => 0.0,
264:       CLKOUT4_PHASE => 0.0,
265:       CLKOUT5_PHASE => 0.0,
266:       CLK_FEEDBACK => "CLKFBOUT", -- Clock source to drive CLKFBIN ("CLKFBOUT" or "CLKOUT0")
267:       COMPENSATION => "SYSTEM_SYNCHRONOUS", -- "SYSTEM_SYNCHRONOUS", "SOURCE_SYNCHRONOUS", "EXTERNAL"
268:       DIVCLK_DIVIDE => 1, -- Division value for all output clocks (1-52)
269:       REF_JITTER => 0.1, -- Reference Clock Jitter in UI (0.000-0.999).
270:       RESET_ON_LOSS_OF_LOCK => FALSE -- Must be set to FALSE
271:    )
272:    port map (
273:       CLKFBOUT => pll_clkfb, -- 1-bit output: PLL_BASE feedback output
274:       -- CLKOUT0 - CLKOUT5: 1-bit (each) output: Clock outputs
275:       CLKOUT0 => clk_48M,
276:       CLKOUT1 => clk_24M,
277:       CLKOUT2 => clk_12M,
278:       CLKOUT3 => clk_8M,
279:       CLKOUT4 => open,
280:       CLKOUT5 => open,
281:       LOCKED => pll_lock, -- 1-bit output: PLL_BASE lock status output
282:       CLKFBIN => pll_clkfb, -- 1-bit input: Feedback clock input
283:       CLKIN => xvclk, -- 1-bit input: Clock input
284:       RST => '0' -- 1-bit input: Reset input
285:    );
286:
287:
288:         pll_lock_n <= not pll_lock;
289:
290:
291:
292:
293:         -- System reset
294:         -- Released synchronously when PLL locks
295:         -- Reasserted asynchronously when PLL loses lock
296:         reset_fdp : fdp
297:         port map (
298:                 c => xvclk,
299:                 d => pll_lock_n,
300:                 q => reset,
```

```
301:                 pre => pll_lock_n
302:         );
303:
304:
305:
306: --      pclk_clkout : cpt_clkout
307: --      generic map (
308: --              CLK_DIV2 => 0
309: --      )
310: --      port map (
311: --              i_clk => clk_24M,
312: --              o_clk => o_ovm_video_miso.pclk
313: --      );
314:
315:
316: --      process(clk_24M)
317: --      begin
318: --              if ( falling_edge(clk_24M) ) then
319: --                      red_pixel <= red_pixel + 1;
320: --                      green_pixel <= green_pixel + 1;
321: --                      blue_pixel <= blue_pixel + 1;
322: --              end if;
323: --      end process;
324:
325:
326: --      process(clk_24M)
327: --      begin
328: --
329: --      end process;
330:
331:
332:
333: -- -----------------------------------------------------
334: -- Video generation
335: -- -----------------------------------------------------
336:
337: --      subpixel <= (H_ACTIVE_WIDTH * v_count + h_count + (v_count mod 4)) mod 4
338: --              when h_active = '1' and v_active = '1'
339: --              else 0;
340:
341:         data <= std_logic_vector(to_unsigned(h_count mod 256, 8))
342:                 when h_active = '1' and v_active = '1'
343:                 else (others => '0');
344:
345: --      process(subpixel)
346: --      begin
347: --              data <= conv_std_logic_vector(character'pos(RGB(subpixel+1)), 8);
348: --      end process;
349:
350:
351:
352:
353:
354:
355: -- -----------------------------------------------------
356: -- Video timing generation
357: -- -----------------------------------------------------
358:
359:         h_counter : cpt_upcounter_testing
360:         generic map (
361:                 INIT => 0
362:         )
363:         port map (
364:                 i_clk => clk_24M,
365:                 i_enable => '1',
366:                 i_lowest => H_FRAME_FIRST,
367:                 i_highest => H_FRAME_LAST,
368:                 i_increment => 1,
369:                 i_clear => reset,
370:                 i_preset => '0',
371:                 o_count => h_count,
372:                 o_carry => open
373:         );
374:
375:         h_active <= '1' when reset = '0' and h_count >= H_ACTIVE_FIRST and h_count <= H_ACTIVE_LAST else '0';
376:         h_sync <= '1' when reset = '0' and h_count >= H_SYNC_FIRST and h_count <= H_SYNC_LAST else '0';
377:
378:         v_counter_enable <= '1' when h_count = H_FRAME_LAST else '0';
379:
380:         v_counter : cpt_upcounter_testing
381:         generic map (
382:                 INIT => 0
383:         )
384:         port map (
385:                 i_clk => clk_24M,
386:                 i_enable => v_counter_enable,
387:                 i_lowest => V_FRAME_FIRST,
388:                 i_highest => V_FRAME_LAST,
389:                 i_increment => 1,
390:                 i_clear => reset,
391:                 i_preset => '0',
392:                 o_count => v_count,
393:                 o_carry => open
394:         );
395:
396:         v_active <= '1' when reset = '0' and v_count >= V_ACTIVE_FIRST and v_count <= V_ACTIVE_LAST else '0';
397:         v_sync <= '1' when reset = '0' and v_count >= V_SYNC_FIRST and v_count <= V_SYNC_LAST else '0';
398:
399:
400:         frame_counter_enable <= '1' when h_count = H_FRAME_LAST and v_count = V_FRAME_LAST else '0';
```

```
401:
402:         frame_counter : cpt_upcounter_testing
403:         generic map (
404:                 INIT => 0
405:         )
406:         port map (
407:                 i_clk => clk_24M,
408:                 i_enable => frame_counter_enable,
409:                 i_lowest => 0,
410:                 i_highest => 2**30-1,
411:                 i_increment => 1,
412:                 i_clear => reset,
413:                 i_preset => '0',
414:                 o_count => frame_count,
415:                 o_carry => open
416:         );
417:
418:         frame_active <= '1' when v_count >= 0 else '0';
419:
420:
421: -- ------------------------------------------------------
422: -- Output buffers
423: -- ------------------------------------------------------
424:
425:         tristate_data <= '0'; --not ovm_reg(OVM_REG0C)(2);
426:         tristate_ctrl <= '0'; --not ovm_reg(OVM_REG0C)(1);
427:
428:         o_ovm_video_miso.data <= data when tristate_data = '0' else (others => 'Z');
429:
430: --      gen_data_obuft :
431: --      for i in 0 to 7 generate
432: --              data_obuft : obuft
433: --              port map (
434: --                      i => data(i),
435: --                      o => o_ovm_video_miso.data(i),
436: --                      t => tristate_data
437: --              );
438: --      end generate;
439:
440:
441:         o_ovm_video_miso.pclk <= clk_24M when tristate_ctrl = '0' else 'Z';
442:
443: --      pclk_obuft : obuft
444: --      port map (
445: --              i => clk_24M,
446: --              o => o_ovm_video_miso.pclk,
447: --              t => tristate_ctrl
448: --      );
449:
450:
451:         o_ovm_video_miso.vsync <= v_sync when tristate_ctrl = '0' else 'Z';
452:
453: --      vsync_obuft : obuft
454: --      port map (
455: --              i => v_sync,
456: --              o => o_ovm_video_miso.vsync,
457: --              t => tristate_ctrl
458: --      );
459:
460:
461:         o_ovm_video_miso.href <= h_active when tristate_ctrl = '0' else 'Z';
462:
463: --      href_obuft : obuft
464: --      port map (
465: --              i => h_active,
466: --              o => o_ovm_video_miso.href,
467: --              t => tristate_ctrl
468: --      );
469:
470:
471:         io_ovm_sccb_bidir.sda <= 'Z';
472:
473: end Behavioral;
474:
```

```
  1:
  2: library IEEE;
  3: use IEEE.STD_LOGIC_1164.ALL;
  4:
  5: library unisim;
  6: use unisim.vcomponents.all;
  7:
  8: entity test_i2c is
  9: end test_i2c;
 10:
 11: architecture Behavioral of test_i2c is
 12:
 13:
 14:         component cpt_i2c is
 15:
 16:                 port (
 17:
 18:                         i_clk : in std_logic;
 19:                         i_enable : in std_logic;
 20:
 21:                         i_scl_clk_div : in integer;
 22:
 23:                         i_addr : in std_logic_vector(6 downto 0);
 24:
 25:                         o_rd_data : out std_logic_vector(7 downto 0);
 26:                         o_rd_data_strobe : out std_logic;
 27:                         i_rd_start : in std_logic;
 28:                         o_rd_done : out std_logic;
 29:
 30:                         i_wr_data_available : in std_logic;
 31:                         i_wr_data : in std_logic_vector(7 downto 0);
 32:                         o_wr_data_strobe : out std_logic;
 33:                         i_wr_start : in std_logic;
 34:                         o_wr_done : out std_logic;
 35:
 36:                         io_i2c_scl : inout std_logic;
 37:                         io_i2c_sda : inout std_logic
 38:
 39:                 );
 40:
 41:         end component;
 42:
 43:
 44:         constant MCU_FREQ : integer := 108_000_000;
 45:         constant I2C_FREQ : integer := 100_000;
 46:
 47:         constant MCU_HALF_PERIOD : time := 1 ns * 1e9 / real(MCU_FREQ);
 48:
 49:         signal clk : std_logic;
 50:         signal enable : std_logic;
 51:
 52:         signal scl_clk_div : integer := MCU_FREQ / (2*8*I2C_FREQ);
 53:
 54:         signal scl : std_logic := 'H';
 55:         signal sda : std_logic := 'H';
 56:
 57:         signal addr : std_logic_vector(6 downto 0) := "1001001";
 58:         signal rd_data : std_logic_vector(7 downto 0);
 59:         signal rd_data_strobe : std_logic;
 60:         signal rd_start : std_logic := '0';
 61:         signal rd_done : std_logic;
 62:
 63:         signal wr_data_available : std_logic := '0';
 64:         signal wr_data : std_logic_vector(7 downto 0) := x"55";
 65:         signal wr_data_strobe : std_logic;
 66:         signal wr_start : std_logic := '0';
 67:         signal wr_done : std_logic;
 68:
 69:
 70: begin
 71:
 72:
 73:         scl_pullup : pullup
 74:         port map (
 75:                 o => scl
 76:         );
 77:
 78:         sda_pullup : pulldown
 79:         port map (
 80:                 o => sda
 81:         );
 82:
 83:         process
 84:         begin
 85:                 enable <= '0';
 86:                 wait for 15 us;
 87:                 enable <= '1';
 88:                 wait;
 89:         end process;
 90:
 91:
 92:         process
 93:         begin
 94:                 wait for 10 us;
 95:                 loop
 96:                         clk <= '1';
 97:                         wait for MCU_HALF_PERIOD;
 98:                         clk <= '0';
 99:                         wait for MCU_HALF_PERIOD;
100:                 end loop;
```

```
101:         end process;
102:
103:
104:
105:
106:         process
107:         begin
108:
109:                 wait for 100 us;
110:                 wait until rising_edge(clk);
111:                 wr_start <= '1';
112:                 wait until wr_done = '0';
113:                 wr_start <= '0';
114:                 wr_data_available <= '1';
115:                 wait until wr_data_strobe = '1';
116:                 wait until wr_data_strobe = '0';
117:                 wr_data_available <= '0';
118:                 wait until wr_done = '1';
119:
120:                 wait for 100 us;
121:                 wait until rising_edge(clk);
122:                 wr_start <= '1';
123:                 wait until wr_done = '0';
124:                 wr_start <= '0';
125:                 wr_data_available <= '1';
126:                 wr_data <= x"55";
127:                 wait until wr_data_strobe = '1';
128:                 wait until wr_data_strobe = '0';
129:                 wr_data_available <= '1';
130:                 wr_data <= x"81";
131:                 wait until wr_data_strobe = '1';
132:                 wait until wr_data_strobe = '0';
133:                 wr_data_available <= '0';
134:                 wait until wr_done = '1';
135:
136:
137:                 wait for 100 us;
138:                 wait until rising_edge(clk);
139:                 rd_start <= '1';
140:                 wait until rd_done = '0';
141:                 rd_start <= '0';
142:                 wait until wr_data_strobe = '1';
143:                 wait until wr_data_strobe = '0';
144:                 wait until rd_done = '1';
145:
146:
147:
148:                 wait;
149:         end process;
150:
151:
152:         i2c : cpt_i2c
153:         port map (
154:                 i_clk => clk,
155:                 i_enable => enable,
156:                 i_scl_clk_div => scl_clk_div,
157:                 i_addr => addr,
158:                 o_rd_data => rd_data,
159:                 o_rd_data_strobe => rd_data_strobe,
160:                 i_rd_start => rd_start,
161:                 o_rd_done => rd_done,
162:                 i_wr_data_available => wr_data_available,
163:                 i_wr_data => wr_data,
164:                 o_wr_data_strobe => wr_data_strobe,
165:                 i_wr_start => wr_start,
166:                 o_wr_done => wr_done,
167:                 io_i2c_scl => scl,
168:                 io_i2c_sda => sda
169:         );
170:
171:
172:
173:
174:
175:
176:
177:
178:
179: end Behavioral;
180:
```

```
  1:
  2: library IEEE;
  3: use IEEE.STD_LOGIC_1164.ALL;
  4:
  5: library unisim;
  6: use unisim.vcomponents.all;
  7:
  8: library cctl;
  9: use cctl.pkg_ovm.all;
 10:
 11: library mcu;
 12: use mcu.pkg_mcu.all;
 13:
 14:
 15:
 16: entity test_iobus_i2c is
 17: end test_iobus_i2c;
 18:
 19: architecture Behavioral of test_iobus_i2c is
 20:
 21:
 22:         component cpt_iobus_i2c is
 23:
 24:                 generic (
 25:                         DEVICE_ID : std_logic_vector(31 downto 0);
 26:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0)
 27:                 );
 28:
 29:                 port (
 30:
 31:                         i_clk : in std_logic;
 32:                         i_enable : in std_logic;
 33:
 34:                         i_iobus_mosi : in typ_mcu_iobus_mosi;
 35:                         o_iobus_miso : out typ_mcu_iobus_miso;
 36:
 37:                         i_scl_clk_div : in integer;
 38:
 39:                         io_scl : inout std_logic;
 40:                         io_sda : inout std_logic
 41:
 42:                 );
 43:
 44:         end component;
 45:
 46:
 47:         signal iobus_mosi : typ_mcu_iobus_mosi := init_mcu_iobus_mosi;
 48:         signal iobus_miso : typ_mcu_iobus_miso := init_mcu_iobus_miso;
 49:
 50:         signal read_data : std_logic_vector(7 downto 0);
 51:
 52:
 53:         constant MCU_FREQ : integer := 108_000_000;
 54:         constant I2C_FREQ : integer := 100_000;
 55:
 56:         constant MCU_HALF_PERIOD : time := 1 ns * 1e9 / real(MCU_FREQ);
 57:
 58:         signal clk : std_logic;
 59:         signal enable : std_logic;
 60:
 61:         signal scl_clk_div : integer := MCU_FREQ / (2*8*I2C_FREQ);
 62:
 63:         signal scl : std_logic := 'H';
 64:         signal sda : std_logic := 'H';
 65:
 66:
 67:
 68:
 69:
 70:
 71: begin
 72:
 73:
 74:
 75:
 76:
 77:         scl_pullup : pullup
 78:         port map (
 79:                 o => scl
 80:         );
 81:
 82:         sda_pullup : pulldown
 83:         port map (
 84:                 o => sda
 85:         );
 86:
 87:         process
 88:         begin
 89:                 enable <= '0';
 90:                 wait for 15 us;
 91:                 enable <= '1';
 92:                 wait;
 93:         end process;
 94:
 95:
 96:         process
 97:         begin
 98:                 wait for 10 us;
 99:                 loop
100:                         clk <= '1';
```

```
101:                        wait for MCU_HALF_PERIOD;
102:                        clk <= '0';
103:                        wait for MCU_HALF_PERIOD;
104:                end loop;
105:        end process;
106:
107:
108:
109:
110:        process
111:        begin
112:
113:                wait for 100 us;
114:                wait until rising_edge(clk);
115:                iobus_mosi.address <= x"F0000055";
116:                iobus_mosi.addr_strobe <= '1';
117:                iobus_mosi.write_data <= x"000000AA";
118:                iobus_mosi.write_strobe <= '1';
119:                wait until rising_edge(clk);
120:                iobus_mosi.address <= x"00000000";
121:                iobus_mosi.addr_strobe <= '0';
122:                iobus_mosi.write_data <= x"00000000";
123:                iobus_mosi.write_strobe <= '0';
124:                wait until iobus_miso.ready = '1';
125:
126:                wait until rising_edge(clk);
127:                iobus_mosi.address <= x"F0000055";
128:                iobus_mosi.addr_strobe <= '1';
129:                iobus_mosi.read_strobe <= '1';
130:                wait until rising_edge(clk);
131:                iobus_mosi.address <= x"00000000";
132:                iobus_mosi.addr_strobe <= '0';
133:                iobus_mosi.read_strobe <= '0';
134:                wait until iobus_miso.ready = '1';
135:                read_data <= iobus_miso.read_data(7 downto 0);
136:
137:                wait;
138:
139:        end process;
140:
141:
142:
143:
144:        iobus_i2c : cpt_iobus_i2c
145:        generic map (
146:                DEVICE_ID => x"F0000000",
147:                DEVICE_ID_MASK => x"FC000000"
148:        )
149:        port map (
150:                i_clk => clk,
151:                i_enable => '1',
152:
153:                i_iobus_mosi => iobus_mosi,
154:                o_iobus_miso => iobus_miso,
155:
156:                i_scl_clk_div => scl_clk_div,
157:
158:                io_scl => scl,
159:                io_sda => sda
160:        );
161:
162:
163:
164: end Behavioral;
165:
```

```
    1:
    2: library IEEE;
    3: use IEEE.STD_LOGIC_1164.ALL;
    4:
    5: library unisim;
    6: use unisim.vcomponents.all;
    7:
    8: library cctl;
    9: use cctl.pkg_ovm.all;
   10:
   11: library mcu;
   12: use mcu.pkg_mcu.all;
   13:
   14:
   15:
   16: entity test_iobus_sccb is
   17: end test_iobus_sccb;
   18:
   19: architecture Behavioral of test_iobus_sccb is
   20:
   21:
   22:         component cpt_iobus_sccb is
   23:
   24:                 generic (
   25:                         DEVICE_ID : std_logic_vector(31 downto 0);
   26:                         DEVICE_ID_MASK : std_logic_vector(31 downto 0)
   27:                 );
   28:
   29:                 port (
   30:
   31:                         i_clk : in std_logic;
   32:                         i_enable : in std_logic;
   33:
   34:                         i_iobus_mosi : in typ_mcu_iobus_mosi;
   35:                         o_iobus_miso : out typ_mcu_iobus_miso;
   36:
   37:                         i_scl_clk_div : in integer;
   38:
   39:                         io_scl : inout std_logic;
   40:                         io_sda : inout std_logic
   41:
   42:                 );
   43:
   44:         end component;
   45:
   46:
   47:         signal iobus_mosi : typ_mcu_iobus_mosi := init_mcu_iobus_mosi;
   48:         signal iobus_miso : typ_mcu_iobus_miso := init_mcu_iobus_miso;
   49:
   50:         signal read_data : std_logic_vector(7 downto 0);
   51:
   52:
   53:         constant MCU_FREQ : integer := 108_000_000;
   54:         constant I2C_FREQ : integer := 100_000;
   55:
   56:         constant MCU_HALF_PERIOD : time := 1 ns * 1e9 / real(MCU_FREQ);
   57:
   58:         signal clk : std_logic;
   59:         signal enable : std_logic;
   60:
   61:         signal scl_clk_div : integer := MCU_FREQ / (2*8*I2C_FREQ);
   62:
   63:         signal scl : std_logic := 'H';
   64:         signal sda : std_logic := 'H';
   65:
   66:
   67:
   68:
   69:
   70:
   71: begin
   72:
   73:
   74:
   75:
   76:
   77:         scl_pullup : pullup
   78:         port map (
   79:                 o => scl
   80:         );
   81:
   82:         sda_pullup : pullup
   83:         port map (
   84:                 o => sda
   85:         );
   86:
   87:         process
   88:         begin
   89:                 enable <= '0';
   90:                 wait for 15 us;
   91:                 enable <= '1';
   92:                 wait;
   93:         end process;
   94:
   95:
   96:         process
   97:         begin
   98:                 wait for 10 us;
   99:                 loop
  100:                         clk <= '1';
```

```
101:                        wait for MCU_HALF_PERIOD;
102:                        clk <= '0';
103:                        wait for MCU_HALF_PERIOD;
104:                end loop;
105:        end process;
106:
107:
108:
109:
110:        process
111:        begin
112:
113:                wait for 100 us;
114:                wait until rising_edge(clk);
115:                iobus_mosi.address <= x"F0000055";
116:                iobus_mosi.addr_strobe <= '1';
117:                iobus_mosi.write_data <= x"000000AA";
118:                iobus_mosi.write_strobe <= '1';
119:                wait until rising_edge(clk);
120:                iobus_mosi.address <= x"00000000";
121:                iobus_mosi.addr_strobe <= '0';
122:                iobus_mosi.write_data <= x"00000000";
123:                iobus_mosi.write_strobe <= '0';
124:                wait until iobus_miso.ready = '1';
125:
126:                wait until rising_edge(clk);
127:                iobus_mosi.address <= x"F0000055";
128:                iobus_mosi.addr_strobe <= '1';
129:                iobus_mosi.read_strobe <= '1';
130:                wait until rising_edge(clk);
131:                iobus_mosi.address <= x"00000000";
132:                iobus_mosi.addr_strobe <= '0';
133:                iobus_mosi.read_strobe <= '0';
134:                wait until iobus_miso.ready = '1';
135:                read_data <= iobus_miso.read_data(7 downto 0);
136:
137:                wait;
138:
139:        end process;
140:
141:
142:
143:
144:        iobus_sccb : cpt_iobus_sccb
145:        generic map (
146:                DEVICE_ID => x"F0000000",
147:                DEVICE_ID_MASK => x"FC000000"
148:        )
149:        port map (
150:                i_clk => clk,
151:                i_enable => '1',
152:
153:                i_iobus_mosi => iobus_mosi,
154:                o_iobus_miso => iobus_miso,
155:
156:                i_scl_clk_div => scl_clk_div,
157:
158:                io_scl => scl,
159:                io_sda => sda
160:        );
161:
162:
163:
164: end Behavioral;
165:
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4: use ieee.std_logic_arith.all;
  5: use ieee.numeric_std.all;
  6:
  7: library unisim;
  8: use unisim.vcomponents.all;
  9:
 10:
 11:
 12: library mctl;
 13: use mctl.pkg_mctl.all;
 14:
 15: library cctl;
 16: use cctl.pkg_cctl.all;
 17:
 18: library cctl;
 19: use cctl.pkg_ovm.all;
 20:
 21: library usb;
 22: use usb.pkg_usb.all;
 23:
 24: library vga;
 25: use vga.pkg_vga.all;
 26:
 27: library util;
 28: use util.pkg_util.all;
 29:
 30:
 31: entity test_quadcam is
 32: end entity test_quadcam;
 33:
 34:
 35: architecture arch of test_quadcam is
 36:
 37:         constant C3_HW_TESTING : string := "FALSE";
 38:         constant C3_SIMULATION : string := "TRUE";
 39:         constant C3_CALIB_SOFT_IP : string := "FALSE";
 40:
 41:
 42:                 -- Duration to stay in reset on startup, in seconds
 43:                 -- External RAM requires 200us min.
 44:         constant STARTUP_RESET_DUR : real := 20.0e-6;
 45:
 46:                 -- Master oscillator frequency, in Hz
 47:         constant SYSTEM_CLOCK_FREQ : real := 24.0e6;
 48:
 49:
 50:         -- Features to include in simulation
 51:         constant INCLUDE_MCU : string := "TRUE";
 52:         constant INCLUDE_MCTL : string := "TRUE";
 53:         constant INCLUDE_MCTL_CHIPSCOPE : string := "FALSE";
 54:         constant INCLUDE_MCTL_TEST : string := "FALSE";
 55:         constant INCLUDE_IOBUS_MPORT : string := "TRUE";
 56:         constant INCLUDE_SCCB : string := "TRUE";
 57:         constant INCLUDE_CCTL : string := "TRUE";
 58:         constant INCLUDE_USB : string := "TRUE";
 59:         constant INCLUDE_VGA : string := "TRUE";
 60:
 61:         constant SMALL_FRAME : string := "FALSE";
 62:
 63:
 64:
 65:         function c3_sim_hw (val1:std_logic_vector( 31 downto 0); val2: std_logic_vector( 31 downto 0) )  return  std_logic_vector is
 66:                 begin
 67:                 if (C3_HW_TESTING = "FALSE") then
 68:                   return val1;
 69:                 else
 70:                   return val2;
 71:                 end if;
 72:         end function;
 73:
 74:
 75:
 76:         component sim_ovm is
 77:                 generic (
 78:                         SMALL_FRAME : string := "FALSE"
 79:                 );
 80:                 port (
 81:                         i_ovm_sccb_mosi : in typ_ovm_sccb_mosi;
 82:                         io_ovm_sccb_bidir : inout typ_ovm_sccb_bidir;
 83:                         o_ovm_video_miso : out typ_ovm_video_miso
 84:                 );
 85:         end component;
 86:
 87:         component top_quadcam is
 88:                 generic (
 89:
 90:                         STARTUP_RESET_DUR : real := 200.0e-6;
 91:                         SYSTEM_CLOCK_FREQ : real := 24.0e6;
 92:
 93:                         INCLUDE_MCU : string;
 94:                         INCLUDE_MCTL : string;
 95:                         INCLUDE_MCTL_CHIPSCOPE : string;
 96:                         INCLUDE_MCTL_TEST : string;
 97:                         INCLUDE_IOBUS_MPORT : string;
 98:                         INCLUDE_SCCB : string;
 99:                         INCLUDE_CCTL : string;
100:                         INCLUDE_USB : string;
```

```
101:                        INCLUDE_VGA : string;
102:
103:                        C3_CALIB_SOFT_IP : string := "FALSE";
104:                        C3_SIMULATION : string := "FALSE";
105:                        C3_HW_TESTING : string := "TRUE"
106:
107:                );
108:
109:                port (
110:
111:                        i_clk_24M : in  std_logic;
112:
113:                        o_mcu_uart_tx : out std_logic;
114:                        i_mcu_uart_rx : in std_logic;
115:
116:                        o_led_addr : out std_logic_vector(2 downto 0);
117:                        i_switch1 : in std_logic;
118:                        i_switch2 : in std_logic;
119:
120:                        mcb3_rzq : inout  std_logic;
121:                        mcb3_cs_n : out std_logic;
122:
123:                        mctl_ram_bidir : inout typ_mctl_ram_bidir;
124:                        mctl_ram_mosi : out typ_mctl_ram_mosi;
125:
126:                        io_debug_ovm0_video_miso : inout typ_ovm_video_miso;
127:                        io_debug_ovm0_sccb_bidir : inout typ_ovm_sccb_bidir;
128:                        o_debug_ovm0_sccb_mosi : inout typ_ovm_sccb_mosi;
129:
130:                        i_ovm1_video_miso : in typ_ovm_video_miso;
131:                        io_ovm1_sccb_bidir : inout typ_ovm_sccb_bidir;
132:                        o_ovm1_sccb_mosi : out typ_ovm_sccb_mosi;
133:
134:                        i_ovm2_video_miso : in typ_ovm_video_miso;
135:                        io_ovm2_sccb_bidir : inout typ_ovm_sccb_bidir;
136:                        o_ovm2_sccb_mosi : out typ_ovm_sccb_mosi;
137:
138:                        i_ovm3_video_miso : in typ_ovm_video_miso;
139:                        io_ovm3_sccb_bidir : inout typ_ovm_sccb_bidir;
140:                        o_ovm3_sccb_mosi : out typ_ovm_sccb_mosi;
141:
142:                        o_vga_mosi : out typ_vga_mosi := init_vga_mosi;
143:
144:                        i_usb_ctrl_miso : in typ_usb_ctrl_miso;
145:                        o_usb_ctrl_mosi : out typ_usb_ctrl_mosi;
146:                        io_usb_data_bidir : inout typ_usb_data_bidir
147:
148:                );
149:        end component;
150:
151:
152:        component lpddr_model_c3 is
153:                port (
154:                        Clk : in std_logic;
155:                        Clk_n : in std_logic;
156:                        Cke : in std_logic;
157:                        Cs_n : in std_logic;
158:                        Ras_n : in std_logic;
159:                        Cas_n : in std_logic;
160:                        We_n : in std_logic;
161:                        Dm : inout std_logic_vector((C3_NUM_DQ_PINS/16) downto 0);
162:                        Ba : in std_logic_vector((C3_MEM_BANKADDR_WIDTH - 1) downto 0);
163:                        Addr : in std_logic_vector((C3_MEM_ADDR_WIDTH  - 1) downto 0);
164:                        Dq : inout std_logic_vector((C3_NUM_DQ_PINS - 1) downto 0);
165:                        Dqs : inout std_logic_vector((C3_NUM_DQ_PINS/16) downto 0)
166:                );
167:        end component;
168:
169:
170:    signal clk_24M : std_logic := '0';
171:
172:         signal mcb3_cs_n : std_logic;
173:
174:         signal mctl_ram_bidir : typ_mctl_ram_bidir;
175:         signal mctl_ram_mosi : typ_mctl_ram_mosi;
176:
177:    signal mcb3_dram_dqs_vector : std_logic_vector(1 downto 0);
178:    signal mcb3_dram_dm_vector : std_logic_vector(1 downto 0);
179:
180:    signal mcb3_command : std_logic_vector(2 downto 0);
181:    signal mcb3_enable1 : std_logic := '0';
182:    signal mcb3_enable2 : std_logic := '0';
183:
184:    signal  mcb3_rzq : std_logic;
185:
186:         signal usb_clkin : std_logic := '0';
187:         signal usb_clkout : std_logic;
188:
189:         signal mcu_uart_tx : std_logic := '0';
190:         signal mcu_uart_rx : std_logic := '0';
191:
192:
193:         signal led_addr : std_logic_vector(2 downto 0);
194:         signal leds : std_logic_vector(7 downto 1);
195:
196:         signal switch1 : std_logic;
197:         signal switch2 : std_logic;
198:
199:         function vector (asi:std_logic) return std_logic_vector is
200:                 variable v : std_logic_vector(0 downto 0) ;
```

```
201:         begin
202:                 v(0) := asi;
203:                 return(v);
204:         end function vector;
205:
206:         constant USB_PERIOD : time := 16.667 ns;
207:
208:
209:         signal debug_ovm0_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
210:         signal ovm1_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
211:         signal ovm2_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
212:         signal ovm3_video_miso : typ_ovm_video_miso := init_ovm_video_miso;
213:
214:
215:         signal debug_ovm0_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
216:         signal ovm1_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
217:         signal ovm2_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
218:         signal ovm3_sccb_bidir : typ_ovm_sccb_bidir := init_ovm_sccb_bidir;
219:
220:         signal debug_ovm0_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
221:         signal ovm1_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
222:         signal ovm2_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
223:         signal ovm3_sccb_mosi : typ_ovm_sccb_mosi := init_ovm_sccb_mosi;
224:
225:         signal vga_mosi : typ_vga_mosi := init_vga_mosi;
226:
227:         signal usb_ctrl_miso : typ_usb_ctrl_miso := init_usb_ctrl_miso;
228:         signal usb_ctrl_mosi : typ_usb_ctrl_mosi := init_usb_ctrl_mosi;
229:         signal usb_data_bidir : typ_usb_data_bidir := init_usb_data_bidir;
230:
231:
232:
233: begin
234:
235:
236:         process
237:         begin
238:                 usb_clkin <= not usb_clkin;
239:                 wait for (USB_PERIOD) / 2;
240:         end process;
241:
242:
243:         -- System clock
244:         process
245:         begin
246:                 clk_24M <= not clk_24M;
247:                 wait for ( 1 sec * period(SYSTEM_CLOCK_FREQ) / 2.0 );
248:         end process;
249:
250:
251:    rzq_pulldown3 : PULLDOWN port map(O => mcb3_rzq);
252:
253:         mcu_uart_rx <= mcu_uart_tx;
254:
255:         quadcam : top_quadcam
256:         generic map (
257:                 STARTUP_RESET_DUR => STARTUP_RESET_DUR,
258:                 SYSTEM_CLOCK_FREQ => SYSTEM_CLOCK_FREQ,
259:
260:                 INCLUDE_MCU => INCLUDE_MCU,
261:                 INCLUDE_MCTL => INCLUDE_MCTL,
262:                 INCLUDE_MCTL_CHIPSCOPE => INCLUDE_MCTL_CHIPSCOPE,
263:                 INCLUDE_MCTL_TEST => INCLUDE_MCTL_TEST,
264:                 INCLUDE_IOBUS_MPORT => INCLUDE_IOBUS_MPORT,
265:                 INCLUDE_SCCB => INCLUDE_SCCB,
266:                 INCLUDE_CCTL => INCLUDE_CCTL,
267:                 INCLUDE_USB => INCLUDE_USB,
268:                 INCLUDE_VGA => INCLUDE_VGA,
269:
270:                 C3_HW_TESTING => C3_HW_TESTING,
271:                 C3_SIMULATION => C3_SIMULATION,
272:                 C3_CALIB_SOFT_IP => C3_CALIB_SOFT_IP
273:         )
274:         port map (
275:
276:                 i_clk_24M => clk_24M,
277:
278:                 o_mcu_uart_tx => mcu_uart_tx,
279:                 i_mcu_uart_rx => mcu_uart_rx,
280:
281:                 o_led_addr => led_addr,
282:                 i_switch1 => switch1,
283:                 i_switch2 => switch2,
284:
285:                 mctl_ram_bidir => mctl_ram_bidir,
286:                 mctl_ram_mosi => mctl_ram_mosi,
287:
288:                 mcb3_cs_n => mcb3_cs_n,
289:                 mcb3_rzq => mcb3_rzq,
290:
291:                 io_debug_ovm0_video_miso => debug_ovm0_video_miso,
292:                 io_debug_ovm0_sccb_bidir => debug_ovm0_sccb_bidir,
293:                 o_debug_ovm0_sccb_mosi => debug_ovm0_sccb_mosi,
294:
295:                 i_ovm1_video_miso => ovm1_video_miso,
296:                 io_ovm1_sccb_bidir => ovm1_sccb_bidir,
297:                 o_ovm1_sccb_mosi => ovm1_sccb_mosi,
298:
299:                 i_ovm2_video_miso => ovm2_video_miso,
300:                 io_ovm2_sccb_bidir => ovm2_sccb_bidir,
```

```
301:                o_ovm2_sccb_mosi => ovm2_sccb_mosi,
302:
303:                i_ovm3_video_miso => ovm3_video_miso,
304:                io_ovm3_sccb_bidir => ovm3_sccb_bidir,
305:                o_ovm3_sccb_mosi => ovm3_sccb_mosi,
306:
307:                o_vga_mosi => vga_mosi,
308:
309:                i_usb_ctrl_miso => usb_ctrl_miso,
310:                o_usb_ctrl_mosi => usb_ctrl_mosi,
311:                io_usb_data_bidir => usb_data_bidir
312:
313:        );
314:
315:
316:        gen_leds :
317:        for i in 1 to 7 generate
318:                leds(i) <= '1' when conv_std_logic_vector(i,3) = led_addr else '0';
319:        end generate;
320:
321:        switch1 <= '0';
322:        switch2 <= '0';
323:
324:
325: -- ============================================================================ --
326: -- Cameras                                                    --
327: -- ============================================================================ --
328:
329:        cam0 : sim_ovm
330:        generic map (
331:                SMALL_FRAME => SMALL_FRAME
332:        )
333:        port map (
334:                i_ovm_sccb_mosi => debug_ovm0_sccb_mosi,
335:                io_ovm_sccb_bidir => debug_ovm0_sccb_bidir,
336:                o_ovm_video_miso => debug_ovm0_video_miso
337:        );
338:
339:        cam1 : sim_ovm
340:        generic map (
341:                SMALL_FRAME => SMALL_FRAME
342:        )
343:        port map (
344:                i_ovm_sccb_mosi => ovm1_sccb_mosi,
345:                io_ovm_sccb_bidir => ovm1_sccb_bidir,
346:                o_ovm_video_miso => ovm1_video_miso
347:        );
348:
349:        cam2 : sim_ovm
350:        generic map (
351:                SMALL_FRAME => SMALL_FRAME
352:        )
353:        port map (
354:                i_ovm_sccb_mosi => ovm2_sccb_mosi,
355:                io_ovm_sccb_bidir => ovm2_sccb_bidir,
356:                o_ovm_video_miso => ovm2_video_miso
357:        );
358:
359:        cam3 : sim_ovm
360:        generic map (
361:                SMALL_FRAME => SMALL_FRAME
362:        )
363:        port map (
364:                i_ovm_sccb_mosi => ovm3_sccb_mosi,
365:                io_ovm_sccb_bidir => ovm3_sccb_bidir,
366:                o_ovm_video_miso => ovm3_video_miso
367:        );
368:
369:
370:
371: -- ============================================================================ --
372: -- LPDDR                                                --
373: -- ============================================================================ --
374:
375:
376:     mcb3_command <= (mctl_ram_mosi.ras_n & mctl_ram_mosi.cas_n & mctl_ram_mosi.we_n);
377:
378:     process(mctl_ram_mosi.ck)
379:     begin
380:       if (rising_edge(mctl_ram_mosi.ck)) then
381:                   if (mcb3_command = "100") then
382:          mcb3_enable2 <= '0';
383:         elsif (mcb3_command = "101") then
384:           mcb3_enable2 <= '1';
385:         else
386:           mcb3_enable2 <= mcb3_enable2;
387:         end if;
388:         mcb3_enable1 <= mcb3_enable2;
389:       end if;
390:     end process;
391:
392:
393:     mcb3_dram_dqs_vector(1 downto 0) <= (mctl_ram_bidir.udqs & mctl_ram_bidir.dqs)
394:                         when (mcb3_enable2 = '0' and mcb3_enable1 = '0') else "ZZ";
395:
396:     mctl_ram_bidir.dqs <= mcb3_dram_dqs_vector(0) when ( mcb3_enable1 = '1') else 'Z';
397:     mctl_ram_bidir.udqs <= mcb3_dram_dqs_vector(1) when (mcb3_enable1 = '1') else 'Z';
398:
399:
400:         mcb3_dram_dm_vector <= (mctl_ram_mosi.udm & mctl_ram_mosi.dm);
```

```
401:
402:         lpddr : lpddr_model_c3
403:         port map(
404:                 Clk => mctl_ram_mosi.ck,
405:                 Clk_n => mctl_ram_mosi.ck_n,
406:                 Cke => mctl_ram_mosi.cke,
407:                 Cs_n => mcb3_cs_n,
408:                 Ras_n => mctl_ram_mosi.ras_n,
409:                 Cas_n => mctl_ram_mosi.cas_n,
410:                 We_n => mctl_ram_mosi.we_n,
411:                 Dm => mcb3_dram_dm_vector ,
412:                 Ba => mctl_ram_mosi.ba,
413:                 Addr => mctl_ram_mosi.a,
414:                 Dq => mctl_ram_bidir.dq,
415:                 Dqs => mcb3_dram_dqs_vector
416:         );
417:
418:
419:
420: end architecture;
```

```
    1:
    2: library ieee;
    3: use ieee.std_logic_1164.all;
    4: use ieee.std_logic_arith.all;
    5: use ieee.std_logic_unsigned.all;
    6: --use ieee.numeric_std.all;
    7:
    8: library unisim;
    9: use unisim.vcomponents.all;
   10:
   11:
   12: library mcu;
   13: use mcu.pkg_mcu.all;
   14:
   15: library mctl;
   16: use mctl.pkg_mctl.all;
   17:
   18: library mctl_test;
   19: use mctl_test.pkg_mctl_test.all;
   20:
   21: library cctl;
   22: use cctl.pkg_cctl.all;
   23: use cctl.pkg_ovm.all;
   24:
   25: library usb;
   26: use usb.pkg_usb.all;
   27:
   28: library vga;
   29: use vga.pkg_vga.all;
   30:
   31: library leds;
   32: use leds.pkg_leds.all;
   33:
   34: library util;
   35: use util.pkg_util.all;
   36:
   37:
   38: entity top_quadcam is
   39:         generic (
   40:                 -- Flag to indicate simulation or hardware
   41:                 -- Set true in testbenches only, and false everywhere else
   42:                 C3_SIMULATION : string := "FALSE";
   43:
   44:
   45:                 ----------------------------------------------------
   46:                 -- Design parameters for hardware implementation
   47:                 ----------------------------------------------------
   48:
   49:                 -- Duration to stay in reset on startup, in seconds
   50:                 -- External RAM requires 200us min.
   51:                 STARTUP_RESET_DUR : real := 200.0e-6;
   52:
   53:                 -- Master oscillator frequency, in Hz
   54:                 SYSTEM_CLOCK_FREQ : real := 24.0e6;
   55:
   56:
   57:                 ----------------------------------------------------
   58:                 -- Features to include in hardware implementation
   59:                 ----------------------------------------------------
   60:
   61:                 -- Microcontroller
   62:                 INCLUDE_MCU : string := "TRUE";
   63:
   64:                 -- RAM controller
   65:                 INCLUDE_MCTL : string := "TRUE";
   66:
   67:                 -- RAM controller debugging
   68:                 INCLUDE_MCTL_CHIPSCOPE : string := "FALSE";
   69:
   70:                 -- RAM traffic generator
   71:                 INCLUDE_MCTL_TEST : string := "FALSE";
   72:
   73:                 -- RAM<-->MCU link
   74:                 INCLUDE_IOBUS_MPORT : string := "TRUE";
   75:
   76:                 -- Camera control bus
   77:                 INCLUDE_SCCB : string := "TRUE";
   78:
   79:                 -- Camera controller
   80:                 INCLUDE_CCTL : string := "TRUE";
   81:
   82:                 -- FTDI USB
   83:                 INCLUDE_USB : string := "TEST";
   84:
   85:                 -- Video output
   86:                 INCLUDE_VGA : string := "TRUE";
   87:
   88:
   89:                 ----------------------------------------------------
   90:                 -- RAM controller parameters
   91:                 ----------------------------------------------------
   92:
   93:                 -- Enable soft calibration logic
   94:                 -- This routine optimizes the RAM I/O by measuring the data channels
   95:                 -- Documentation says it is not supported on LPDDR designs
   96:                 C3_CALIB_SOFT_IP : string := "FALSE";
   97:
   98:                 -- Determines the address space accessed by the traffic generator
   99:                 -- # = FALSE, Smaller address space,
  100:                 -- # = TRUE, Large address space.
```

```
101:                C3_HW_TESTING : string := "TRUE"
102:
103:
104:                -- # = 1, Enable debug signals/controls
105:                --   = 0, Disable debug signals/controls
106:                --DEBUG_EN : integer := 1
107:        );
108:        port (
109:                --o_ram_error : out std_logic;
110:                o_mcu_uart_tx : out std_logic;
111:                i_mcu_uart_rx : in std_logic;
112:                --o_calib_done : out std_logic;
113:                --o_error : out std_logic;
114:                i_clk_24m : in std_logic;
115:
116:                o_error_led_n : out std_logic;
117:                --i_reset : in  std_logic;
118:
119:                o_led_addr : out std_logic_vector(2 downto 0);
120:
121:
122:                i_switch1 : in std_logic;
123:                i_switch2 : in std_logic;
124:
125:                mcb3_rzq : inout std_logic;
126:                mcb3_cs_n : out std_logic;
127:
128:                mctl_ram_bidir : inout typ_mctl_ram_bidir;
129:                mctl_ram_mosi : out typ_mctl_ram_mosi;
130:
131:                i_ovm0_video_miso : inout typ_ovm_video_miso;
132:                io_ovm0_sccb_bidir : inout typ_ovm_sccb_bidir;
133:                o_ovm0_sccb_mosi : inout typ_ovm_sccb_mosi;
134:
135:                i_ovm1_video_miso : in typ_ovm_video_miso;
136:                io_ovm1_sccb_bidir : inout typ_ovm_sccb_bidir;
137:                o_ovm1_sccb_mosi : out typ_ovm_sccb_mosi;
138:
139:                i_ovm2_video_miso : in typ_ovm_video_miso;
140:                io_ovm2_sccb_bidir : inout typ_ovm_sccb_bidir;
141:                o_ovm2_sccb_mosi : out typ_ovm_sccb_mosi;
142:
143:                i_ovm3_video_miso : in typ_ovm_video_miso;
144:                io_ovm3_sccb_bidir : inout typ_ovm_sccb_bidir;
145:                o_ovm3_sccb_mosi : out typ_ovm_sccb_mosi;
146:
147:                o_vga_mosi : out typ_vga_mosi := init_vga_mosi;
148:
149:                i_usb_ctrl_miso : in typ_usb_ctrl_miso;
150:                o_usb_ctrl_mosi : out typ_usb_ctrl_mosi;
151:                io_usb_data_bidir : inout typ_usb_data_bidir;
152:
153:                o_wifi_txd : out std_logic;
154:                io_wifi_rxd : inout std_logic;
155:                o_wifi_rst : out std_logic;
156:                io_wifi_gpio0 : inout std_logic;
157:                io_wifi_gpio2 : inout std_logic;
158:                o_wifi_ch_pd : out std_logic;
159:
160:                o_dummy_bank0_topright : out std_logic;
161:                o_dummy_bank0_topleft : out std_logic;
162:                o_dummy_bank1_righttop : out std_logic;
163:                o_dummy_bank1_rightbottom : out std_logic;
164:                o_dummy_bank2_bottomleft : out std_logic;
165:                o_dummy_bank2_bottomright : out std_logic;
166:                o_dummy_bank3_lefttop : out std_logic;
167:                o_dummy_bank3_leftbottom : out std_logic
168:        );
169: end top_quadcam;
170:
171: architecture arch of top_quadcam is
172:
173:
174:
175:    COMPONENT cpt_ovm_bram
176:     PORT(
177:                        i_pclk : in std_logic;
178:                        i_vsync : in std_logic;
179:                        i_href : in std_logic;
180:                        i_data : in std_logic_vector (7 downto 0);
181:                        i_reset : in std_logic;
182:
183:                        o_rd_data : out std_logic_vector(31 downto 0);
184:                        o_frame_number : out integer range 0 to 3;
185:                        o_line_number : out integer range 0 to 2047;
186:
187:                        o_words_read: out integer range 0 to 511;
188:                        i_burst_length : std_logic_vector(5 downto 0);
189:                        o_burst_available : out std_logic;
190:                        o_collision : out std_logic;
191:
192:                        i_clk : in std_logic;
193:                        i_rd_enable : in std_logic
194:        );
195:     END COMPONENT;
196:
197:
198:     COMPONENT cpt_ovm_mux
199:     PORT(
200:          i_clk : IN  std_logic;
```

```
201:          i_reset : IN  std_logic;
202:          i0_frame_count : IN  integer range 0 to 3;
203:          i1_frame_count : IN  integer range 0 to 3;
204:          i2_frame_count : IN  integer range 0 to 3;
205:          i3_frame_count : IN  integer range 0 to 3;
206:          i_frame_addr0 : IN  std_logic_vector(28 downto 0);
207:          i_frame_addr1 : IN  std_logic_vector(28 downto 0);
208:          i_frame_addr2 : IN  std_logic_vector(28 downto 0);
209:          i_frame_addr3 : IN  std_logic_vector(28 downto 0);
210:          i0_line_offset : IN  integer range 0 to 8191;
211:          i1_line_offset : IN  integer range 0 to 8191;
212:          i2_line_offset : IN  integer range 0 to 8191;
213:          i3_line_offset : IN  integer range 0 to 8191;
214:          i0_words_read : IN  integer range 0 to 511;
215:          i1_words_read : IN  integer range 0 to 511;
216:          i2_words_read : IN  integer range 0 to 511;
217:          i3_words_read : IN  integer range 0 to 511;
218:          i0_line_count : IN  integer range 0 to 511;
219:          i1_line_count : IN  integer range 0 to 511;
220:          i2_line_count : IN  integer range 0 to 511;
221:          i3_line_count : IN  integer range 0 to 511;
222:          i0_rd_data : IN  std_logic_vector(31 downto 0);
223:          i1_rd_data : IN  std_logic_vector(31 downto 0);
224:          i2_rd_data : IN  std_logic_vector(31 downto 0);
225:          i3_rd_data : IN  std_logic_vector(31 downto 0);
226:          i0_burst_available : IN  std_logic;
227:          i1_burst_available : IN  std_logic;
228:          i2_burst_available : IN  std_logic;
229:          i3_burst_available : IN  std_logic;
230:          o0_rd_enable : OUT  std_logic;
231:          o1_rd_enable : OUT  std_logic;
232:          o2_rd_enable : OUT  std_logic;
233:          o3_rd_enable : OUT  std_logic;
234:          i_burst_length : IN  std_logic_vector(5 downto 0);
235:          o_mport_miso : OUT  typ_mctl_mport_miso;
236:          i_mport_mosi : IN  typ_mctl_mport_mosi
237:         );
238:     END COMPONENT;
239:
240:
241:
242:
243:
244: -- ============================================================================= --
245: -- Signal Declarations                                                         --
246: -- ============================================================================= --
247:
248:         signal clk_24m : std_logic := '0';
249:
250:         --signal ovm0_video_miso : typ_ovm_video_miso;
251:         --signal ovm0_sccb_bidir : typ_ovm_sccb_bidir;
252:         --signal ovm0_sccb_mosi : typ_ovm_sccb_mosi;
253:
254:
255:
256: --      signal debug_data : std_logic_vector(7 downto 0);
257: --      signal debug_href : std_logic;
258: --      signal debug_vsync : std_logic;
259: --      signal debug_scl : std_logic;
260: --      --signal debug_sda : std_logic;
261: --      signal debug_pclk : std_logic;
262: --
263: --      signal debug0_data : std_logic_vector(7 downto 0);
264: --      signal debug1_data : std_logic_vector(7 downto 0);
265: --      signal debug2_data : std_logic_vector(7 downto 0);
266: --      signal debug3_data : std_logic_vector(7 downto 0);
267: --      signal debug4_data : std_logic_vector(7 downto 0);
268: --      signal debug5_data : std_logic_vector(7 downto 0);
269: --      signal debug6_data : std_logic_vector(7 downto 0);
270: --      signal debug7_data : std_logic_vector(7 downto 0);
271: --      signal debug8_data : std_logic_vector(7 downto 0);
272: --      signal debug9_data : std_logic_vector(7 downto 0);
273: --      signal debugA_data : std_logic_vector(7 downto 0);
274: --      signal debugB_data : std_logic_vector(7 downto 0);
275: --      signal debugC_data : std_logic_vector(7 downto 0);
276: --      signal debugD_data : std_logic_vector(7 downto 0);
277: --      signal debugE_data : std_logic_vector(7 downto 0);
278: --      signal debugF_data : std_logic_vector(7 downto 0);
279: --      signal debug_hrefs : std_logic_vector(15 downto 0);
280: --      signal debug_vsyncs : std_logic_vector(15 downto 0);
281: --      signal debug_scls : std_logic_vector(15 downto 0);
282: --      --signal debug_sda : std_logic;
283: --      signal debug_pclks : std_logic_vector(15 downto 0);
284: --
285: --      signal debug_output_enable : std_logic;
286: --      signal debug_output_enable_n : std_logic;
287: --
288: --      signal debug_src : integer range 0 to 15;
289: --
290: --      signal debug : std_logic_vector(31 downto 0);
291: --
292: --      signal probe_src : integer range 0 to 15;
293: --      signal probe_latch_div : integer;
294: --      signal probe_latch_pgate : std_logic;
295: --
296: --      signal probe_value : std_logic;
297: --      --signal probe_value_n : std_logic;
298: --
299: --      signal probe_value_gated_n : std_logic;
300: --      signal probe_value_gated_p : std_logic;
```

```
301: --
302: --      signal probe_enable : std_logic;
303: --      signal probe_enable_n : std_logic;
304: --
305: --      signal probe_clear : std_logic;
306: --
307: --      signal probe_low_count : integer;
308: --      signal probe_high_count : integer;
309:
310:
311:
312:
313:
314:         signal ovm0_video_miso : typ_ovm_video_miso;
315: --      signal ovm1_sccb_bidir : typ_ovm_sccb_bidir;
316:         signal ovm0_sccb_mosi : typ_ovm_sccb_mosi;
317:
318:         signal ovm1_video_miso : typ_ovm_video_miso;
319: --      signal ovm1_sccb_bidir : typ_ovm_sccb_bidir;
320:         signal ovm1_sccb_mosi : typ_ovm_sccb_mosi;
321:
322:         signal ovm2_video_miso : typ_ovm_video_miso;
323: --      signal ovm2_sccb_bidir : typ_ovm_sccb_bidir;
324:         signal ovm2_sccb_mosi : typ_ovm_sccb_mosi;
325:
326:         signal ovm3_video_miso : typ_ovm_video_miso;
327: --      signal ovm3_sccb_bidir : typ_ovm_sccb_bidir;
328:         signal ovm3_sccb_mosi : typ_ovm_sccb_mosi;
329:
330:    signal burst_length : std_logic_vector(5 downto 0) := conv_std_logic_vector(15,6);
331:
332:         signal ovm_frame_addr0 : std_logic_vector(28 downto 0) := x"0000000" & "0";
333:    signal ovm_frame_addr1 : std_logic_vector(28 downto 0) := x"1000000" & "0";
334:    signal ovm_frame_addr2 : std_logic_vector(28 downto 0) := x"2000000" & "0";
335:    signal ovm_frame_addr3 : std_logic_vector(28 downto 0) := x"3000000" & "0";
336:
337:         signal vga_frame_addr0 : std_logic_vector(28 downto 0) := x"0000000" & "0";
338:    signal vga_frame_addr1 : std_logic_vector(28 downto 0) := x"1000000" & "0";
339:    signal vga_frame_addr2 : std_logic_vector(28 downto 0) := x"2000000" & "0";
340:    signal vga_frame_addr3 : std_logic_vector(28 downto 0) := x"3000000" & "0";
341:
342:    signal ovm0_line_offset : integer range 0 to 8191 := 0;
343:    signal ovm1_line_offset : integer range 0 to 8191 := 1024;
344:    signal ovm2_line_offset : integer range 0 to 8191 := 1024;
345:    signal ovm3_line_offset : integer range 0 to 8191 := 640;
346:
347:
348:
349:
350:
351:         signal ovm_mux_enable : std_logic;
352:         signal ovm_mux_reset : std_logic;
353:         signal ovm_bram_enable : std_logic_vector(3 downto 0);
354:
355:         signal ovm0_bram_reset : std_logic;
356:    signal ovm0_bram_rd_enable : std_logic := '0';
357:    signal ovm0_bram_rd_data : std_logic_vector(31 downto 0);
358:    signal ovm0_bram_frame_number : integer range 0 to 3;
359:    signal ovm0_bram_line_number : integer range 0 to 2047;
360:    signal ovm0_bram_words_read : integer range 0 to 511;
361:    signal ovm0_bram_burst_available : std_logic;
362:
363:         signal ovm1_bram_reset : std_logic;
364:    signal ovm1_bram_rd_enable : std_logic := '0';
365:    signal ovm1_bram_rd_data : std_logic_vector(31 downto 0);
366:    signal ovm1_bram_frame_number : integer range 0 to 3;
367:    signal ovm1_bram_line_number : integer range 0 to 2047;
368:    signal ovm1_bram_words_read : integer range 0 to 511;
369:    signal ovm1_bram_burst_available : std_logic;
370:
371:         signal ovm2_bram_reset : std_logic;
372:    signal ovm2_bram_rd_enable : std_logic := '0';
373:    signal ovm2_bram_rd_data : std_logic_vector(31 downto 0);
374:    signal ovm2_bram_frame_number : integer range 0 to 3;
375:    signal ovm2_bram_line_number : integer range 0 to 2047;
376:    signal ovm2_bram_words_read : integer range 0 to 511;
377:    signal ovm2_bram_burst_available : std_logic;
378:
379:         signal ovm3_bram_reset : std_logic;
380:    signal ovm3_bram_rd_enable : std_logic := '0';
381:    signal ovm3_bram_rd_data : std_logic_vector(31 downto 0);
382:    signal ovm3_bram_frame_number : integer range 0 to 3;
383:    signal ovm3_bram_line_number : integer range 0 to 2047;
384:    signal ovm3_bram_words_read : integer range 0 to 511;
385:    signal ovm3_bram_burst_available : std_logic;
386:
387:    signal o0_collision : std_logic;
388:    signal o1_collision : std_logic;
389:    signal o2_collision : std_logic;
390:    signal o3_collision : std_logic;
391:
392:
393:
394:         signal vga_mosi : typ_vga_mosi;
395:
396:
397:
398:         signal mcu_gpi : typ_mcu_word_array := (others => (others => '0'));
399:         signal mcu_gpo : typ_mcu_word_array;
400:
```

```
401:
402:
403:         signal mcu_intc_interrupt : std_logic_vector(7 downto 0);
404:         signal mcu_intc_irq : std_logic_vector(31 downto 0);
405:
406:         signal user_reset : std_logic;
407:
408:         signal ram_status : std_logic_vector(1 downto 0);
409:
410:         signal  c3_error  : std_logic := '0';
411:         signal  c3_calib_done : std_logic := '0';
412:         signal  c3_error_status : std_logic_vector(127 downto 0) := (others => '0');
413:
414:         signal  clk_108 : std_logic := '0';
415:         signal  clk_108_n : std_logic := '1';
416:
417:         signal  c3_clk0 : std_logic := '0';
418:         signal  c3_cmp_error : std_logic := '0';
419:         signal  c3_vio_modify_enable : std_logic := '0';
420:         signal  c3_vio_data_mode_value : std_logic_vector(2 downto 0) := (others => '0');
421:         signal  c3_vio_addr_mode_value : std_logic_vector(2 downto 0) := (others => '0');
422:
423:
424:         signal  mctl_test_mport0_enabled : std_logic := '1';
425:         signal  mctl_test_mport1_enabled : std_logic := '1';
426:         signal  mctl_test_mport2_enabled : std_logic := '1';
427:         signal  mctl_test_mport3_enabled : std_logic := '0';
428:
429:
430:         signal mctl_test_mport3_mosi : typ_mctl_mport_mosi := init_mctl_mport_mosi;
431:         signal mctl_test_mport3_miso : typ_mctl_mport_miso := init_mctl_mport_miso;
432:
433:
434:
435:         signal  c3_selfrefresh_enter                    : std_logic := '0';
436:         signal  c3_selfrefresh_mode                     : std_logic := '0';
437:
438:
439:
440:         signal dummy_usb_data_bidir : typ_usb_data_bidir;
441:
442:
443:         -- Memory controller signals
444:         signal mctl_mport0_mosi : typ_mctl_mport_mosi := init_mctl_mport_mosi;
445:         signal mctl_mport0_miso : typ_mctl_mport_miso := init_mctl_mport_miso;
446:
447:         signal mctl_mport1_mosi : typ_mctl_mport_mosi := init_mctl_mport_mosi;
448:         signal mctl_mport1_miso : typ_mctl_mport_miso := init_mctl_mport_miso;
449:
450:         signal mctl_mport2_mosi : typ_mctl_mport_mosi := init_mctl_mport_mosi;
451:         signal mctl_mport2_miso : typ_mctl_mport_miso := init_mctl_mport_miso;
452:
453:         signal mctl_mport3_mosi : typ_mctl_mport_mosi := init_mctl_mport_mosi;
454:         signal mctl_mport3_miso : typ_mctl_mport_miso := init_mctl_mport_miso;
455:
456:
457:         -- UART signals
458:         signal uart_txd : std_logic;
459:         signal uart_rxd : std_logic;
460:
461:
462:         -- WiFi signals
463:         signal wifi_txd : std_logic;
464:         --signal wifi_rxd : std_logic;
465:         signal wifi_rst : std_logic;
466:         signal wifi_gpio0 : std_logic;
467:         signal wifi_gpio2 : std_logic;
468:         signal wifi_ch_pd : std_logic;
469:
470:
471:         -- LED signals
472:         signal error_led : std_logic;
473:
474:         signal error_led0 : std_logic;
475:         signal error_led1 : std_logic;
476:         signal error_led2 : std_logic;
477:         signal error_led3 : std_logic;
478:
479:         signal error_led_src : integer range 0 to 3;--std_logic_vector(31 downto 0);
480:
481:         signal leds : std_logic_vector(7 downto 1);
482:
483:         signal leds0 : std_logic_vector(7 downto 1);
484:         signal leds1 : std_logic_vector(7 downto 1);
485:         signal leds2 : std_logic_vector(7 downto 1);
486:         signal leds3 : std_logic_vector(7 downto 1);
487:         signal leds4 : std_logic_vector(7 downto 1);
488:         signal leds5 : std_logic_vector(7 downto 1);
489:         signal leds6 : std_logic_vector(7 downto 1);
490:         signal leds7 : std_logic_vector(7 downto 1);
491:         signal leds8 : std_logic_vector(7 downto 1);
492:         signal leds9 : std_logic_vector(7 downto 1);
493:         signal ledsA : std_logic_vector(7 downto 1);
494:         signal ledsB : std_logic_vector(7 downto 1);
495:         signal ledsC : std_logic_vector(7 downto 1);
496:         signal ledsD : std_logic_vector(7 downto 1);
497:         signal ledsE : std_logic_vector(7 downto 1);
498:         signal ledsF : std_logic_vector(7 downto 1);
499:
500:         signal leds_src : std_logic_vector(31 downto 0);
```

```
501:
502:         signal ledclk_div : integer;
503:         signal led_latch_div : integer;
504:
505:
506:         -- Flash signals
507:         signal flash_clk_div : integer;
508:         signal flash_on : integer;
509:         signal flash_max : integer;
510:         signal flash_count : integer;
511:         signal flash_value : std_logic;
512:         signal flash_clk_pgate : std_logic;
513:
514:
515:         -- VGA signals
516:         signal vga_mosi_fixed : typ_vga_mosi;
517:         signal vga_mosi_test : typ_vga_mosi;
518:
519:         signal vga_mport_mosi : typ_mctl_mport_mosi;
520:         signal vga_mport_miso : typ_mctl_mport_miso;
521:
522:         signal vga_src : integer;
523:
524:
525:         signal vga_fixed_enable : std_logic;
526:
527:         signal vga_test_mode : std_logic;
528:         signal vga_test_enable : std_logic;
529:
530:         -- vga test2
531:         signal vga_enable : std_logic := '0';
532:         signal line_start : std_logic;
533:         signal pixel_number : integer range -2048 to 2047;
534:         signal line_number : integer range -1024 to 1023;
535:         signal frame_number : integer range 0 to 3;
536:         signal linebuf_data : std_logic_vector(15 downto 0);
537:         signal vga_mid_line_offset : integer range -(2**24) to (2**24)-1 := 0;
538:
539:
540:         function vector (asi:std_logic) return std_logic_vector is
541:                 variable v : std_logic_vector(0 downto 0);
542:         begin
543:                 v(0) := asi;
544:         return(v);
545:         end function vector;
546:
547:
548:         function or_slv (v:std_logic_vector) return std_logic is
549:                 variable o : std_logic;
550:         begin
551:                 o := '0';
552:                 for i in v'range loop
553:                         o := o or v(i);
554:                 end loop;
555:         return o;
556:         end function;
557:
558:
559:         -- Reset signals
560:         signal reset : std_logic := '1';
561:         signal reset_n : std_logic := '0';
562:         signal reset_long : std_logic := '1';
563:         signal reset_long_n : std_logic := '0';
564:
565: begin
566:
567:         i_clk_24m_ibufg : ibufg
568:         port map (
569:                 I  => i_clk_24m,
570:                 O  => clk_24m
571:         );
572:
573:         mcb3_cs_n <= '0';
574:         user_reset <= i_switch2;        -- Physical SW2 on board
575:
576:         process(clk_24m)
577:         begin
578:                 if ( rising_edge(clk_24m) ) then
579:                         reset <= user_reset;
580:                 end if;
581:         end process;
582:
583:         -- Reset counter
584:         -- Holds reset high for the duration specified by STARTUP_RESET_DUR
585:         reset_counter : cpt_upcounter
586:         generic map (
587:                 INIT => 1
588:         )
589:         port map (
590:                 i_clk => clk_24m,
591:                 i_enable => reset_long, -- Self-disable. reset must be initialized to '1'
592:                 i_lowest => 1,
593:                 i_highest => cycles_f(STARTUP_RESET_DUR, SYSTEM_CLOCK_FREQ),
594:                 i_increment => 1,
595:                 i_clear => reset,
596:                 i_preset => '0',
597:                 o_count => open,
598:                 o_carry => reset_long_n
599:         );
600:
```

```
601:
602:         reset_long <= not reset_long_n;
603:
604:
605:
606:         -- TODO: create ram_status vector, use here and as LED src
607:         mcu_gpi(GPIO_RAM_STATUS)(1 downto 0) <= ram_status;
608:         mcu_gpi(GPIO_RAM_ERROR_STATUS3) <= c3_error_status(127 downto 96);
609:         mcu_gpi(GPIO_RAM_ERROR_STATUS2) <= c3_error_status(95 downto 64);
610:         mcu_gpi(GPIO_RAM_ERROR_STATUS1) <= c3_error_status(63 downto 32);
611:         mcu_gpi(GPIO_RAM_ERROR_STATUS0) <= c3_error_status(31 downto 0);
612:
613:
614:
615: --      debug0_data <= mcu_gpo(GPIO_DEBUG0)(7 downto 0);
616: --      mcu_gpi(GPIO_DEBUG0)(7 downto 0) <= debug0_data;
617:
618: --      debug_pclks(0) <= mcu_gpo(GPIO_DEBUG0)(8);
619: --      mcu_gpi(GPIO_DEBUG0)(8) <= debug_pclks(0);
620:
621: --      debug_hrefs(0) <= mcu_gpo(GPIO_DEBUG0)(9);
622: --      mcu_gpi(GPIO_DEBUG0)(9) <= debug_hrefs(0);
623:
624: --      debug_vsyncs(0) <= mcu_gpo(GPIO_DEBUG0)(10);
625: --      mcu_gpi(GPIO_DEBUG0)(10) <= debug_vsyncs(0);
626:
627:
628:         --debug4_data(0) <= wifi_txd;
629:         --debug4_data(1) <= wifi_rxd;
630:         --debug4_data(2) <= wifi_ch_pd;
631:         --debug4_data(3) <= wifi_rst;
632:         --debug4_data(4) <= io_wifi_gpio0;
633:         --debug4_data(5) <= io_wifi_gpio2;
634:
635:         --debug4_data(6) <= uart_txd;
636:         --debug4_data(7) <= uart_rxd;
637:
638:
639: --      debug7_data(3 downto 0) <= vga_mosi.red;
640: --      debug7_data(7 downto 4) <= vga_mosi.green;
641: --      debug_hrefs(7) <= vga_mosi.hsync;
642: --      debug_vsyncs(7) <= vga_mosi.vsync;
643: --
644: --      debug8_data(3 downto 0) <= vga_mosi.green;
645: --      debug8_data(7 downto 4) <= vga_mosi.blue;
646: --      debug_hrefs(8) <= vga_mosi.hsync;
647: --      debug_vsyncs(8) <= vga_mosi.vsync;
648: --
649: --      debug9_data(3 downto 0) <= vga_mosi.red;
650: --      debug9_data(7 downto 4) <= vga_mosi.blue;
651: --      debug_hrefs(9) <= vga_mosi.hsync;
652: --      debug_vsyncs(9) <= vga_mosi.vsync;
653:         --debug_scls(9) <= vga_mosi.blue(1);
654:         --debug_sda(9) <= vga_mosi.blue(0);
655:
656:
657: --      debugA_data <= ovm0_video_miso.data;
658: --      debug_pclks(16#A#) <= ovm0_video_miso.pclk;
659: --      debug_hrefs(16#A#) <= ovm0_video_miso.href;
660: --      debug_vsyncs(16#A#) <= ovm0_video_miso.vsync;
661: --
662: --      debugB_data <= i_ovm1_video_miso.data;
663: --      debug_pclks(16#B#) <= i_ovm1_video_miso.pclk;
664: --      debug_hrefs(16#B#) <= i_ovm1_video_miso.href;
665: --      debug_vsyncs(16#B#) <= i_ovm1_video_miso.vsync;
666: --
667: --      debugC_data <= i_ovm2_video_miso.data;
668: --      debug_pclks(16#C#) <= i_ovm2_video_miso.pclk;
669: --      debug_hrefs(16#C#) <= i_ovm2_video_miso.href;
670: --      debug_vsyncs(16#C#) <= i_ovm2_video_miso.vsync;
671: --
672: --      debugD_data <= i_ovm3_video_miso.data;
673: --      debug_pclks(16#D#) <= i_ovm3_video_miso.pclk;
674: --      debug_hrefs(16#D#) <= i_ovm3_video_miso.href;
675: --      debug_vsyncs(16#D#) <= i_ovm3_video_miso.vsync;
676:
677:
678: --      with debug_src select debug_data <=
679: --              debug0_data when 16#0#,
680: --              debug1_data when 16#1#,
681: --              debug2_data when 16#2#,
682: --              debug3_data when 16#3#,
683: --              debug4_data when 16#4#,
684: --              debug5_data when 16#5#,
685: --              debug6_data when 16#6#,
686: --              debug7_data when 16#7#,
687: --              debug8_data when 16#8#,
688: --              debug9_data when 16#9#,
689: --              debugA_data when 16#A#,
690: --              debugB_data when 16#B#,
691: --              debugC_data when 16#C#,
692: --              debugD_data when 16#D#,
693: --              debugE_data when 16#E#,
694: --              debugF_data when 16#F#,
695: --              (others => '1') when others;
696: --
697: --
698: --      debug_pclk <= debug_pclks(debug_src);
699: --      debug_href <= debug_hrefs(debug_src);
700: --      debug_vsync <= debug_vsyncs(debug_src);
```

```
701:
702:         -- The MCU can read the debug port via the
703:         -- DEBUG register (whether output enabled or not)
704: --      debug(7 downto 0) <= debug_data;
705: --      debug(8) <= debug_pclk;
706: --      debug(9) <= debug_href;
707: --      debug(10) <= debug_vsync;
708: --
709: --      mcu_gpi(GPIO_DEBUG) <= debug;
710:
711:
712:
713: --      probe_enable <= mcu_gpo(GPIO_PROBE_ENABLE)(0);
714: --      mcu_gpi(GPIO_PROBE_ENABLE)(0) <= probe_enable;
715: --
716: --      probe_clear <= mcu_gpo(GPIO_PROBE_CLEAR)(0);
717: --      mcu_gpi(GPIO_PROBE_CLEAR)(0) <= probe_clear;
718: --
719: --
720: --      probe_src <= conv_integer(mcu_gpo(GPIO_PROBE_SRC));
721: --      mcu_gpi(GPIO_PROBE_SRC) <= conv_std_logic_vector(probe_src, 32);
722: --
723: --      probe_latch_div <= conv_integer(mcu_gpo(GPIO_PROBE_LATCH_DIV));
724: --      mcu_gpi(GPIO_PROBE_SRC) <= conv_std_logic_vector(probe_src, 32);
725: --
726: --
727: --      probe_latch_gate : cpt_clk_gate
728: --      port map (
729: --              i_clk => c3_clk0,
730: --              i_enable => '1',
731: --              i_div => probe_latch_div,
732: --              o_clk_pgate => probe_latch_pgate,
733: --              o_clk_ngate => open
734: --      );
735: --
736: --
737: --      process(c3_clk0)
738: --      begin
739: --              if ( rising_edge(c3_clk0) and probe_latch_pgate = '1' ) then
740: --                      probe_value <= debug(probe_src);
741: --              end if;
742: --      end process;
743: --
744: --      probe_value <= debug(probe_src);
745: --
746: --      probe_value_gated_p <= probe_value and probe_latch_pgate;
747: --      probe_value_gated_n <= not probe_value and probe_latch_pgate;
748: --
749: --
750: --      probe_low_counter : cpt_upcounter
751: --      generic map (
752: --              INIT => 0
753: --      )
754: --      port map (
755: --              i_clk => c3_clk0,
756: --              i_enable => probe_value_gated_n,
757: --              i_lowest => 0,
758: --              i_highest => 2**30-1,
759: --              i_increment => 1,
760: --              i_clear => probe_clear,
761: --              i_preset => '0',
762: --              o_count => probe_low_count,
763: --              o_carry => open
764: --      );
765: --
766: --      process(c3_clk0)
767: --      begin
768: --              if ( rising_edge(c3_clk0) ) then
769: --                      mcu_gpi(GPIO_PROBE_LOW) <= conv_std_logic_vector(probe_low_count, 32);
770: --              end if;
771: --      end process;
772: --
773: --
774: --      probe_high_counter : cpt_upcounter
775: --      generic map (
776: --              INIT => 0
777: --      )
778: --      port map (
779: --              i_clk => c3_clk0,
780: --              i_enable => probe_value_gated_p,
781: --              i_lowest => 0,
782: --              i_highest => 2**30-1,
783: --              i_increment => 1,
784: --              i_clear => probe_clear,
785: --              i_preset => '0',
786: --              o_count => probe_high_count,
787: --              o_carry => open
788: --      );
789: --
790: --
791: --      process(c3_clk0)
792: --      begin
793: --              if ( rising_edge(c3_clk0) ) then
794: --                      mcu_gpi(GPIO_PROBE_HIGH) <= conv_std_logic_vector(probe_high_count, 32);
795: --              end if;
796: --      end process;
797:
798:
799: --      o_debug_ovm0_sccb_mosi.pwdn <= ovm0_sccb_mosi.pwdn;
800: --      o_debug_ovm0_sccb_mosi.xvclk <= ovm0_sccb_mosi.xvclk;
```

```
801:
802:
803:         -- TODO: add logic to handle SCL and SDA debug output plus normal I/O
804:         --o_debug_ovm0_sccb_mosi.scl <= ovm0_sccb_mosi.scl;
805:         --io_debug_ovm0_sccb_mosi.sda <= debug_sda when debug_output_enable = '1' else ovm0_sccb_bidir.sda;
806:         --ovm0_sccb_bidir.sda <= io_debug_ovm0_sccb_mosi.sda;
807:
808:
809: --      debug_output_enable_n <= not debug_output_enable;
810: --
811: --
812: --      gen_debug_data_iobuf :
813: --      for i in 0 to 7 generate
814: --      begin
815: --              debug_data_iobuf : iobuf
816: --              generic map (
817: --                      drive => 2,
818: --                      iostandard => "lvcmos18",
819: --                      slew => "slow")
820: --              port map (
821: --                      o => ovm0_video_miso.data(i),
822: --                      io => io_debug_ovm0_video_miso.data(i),
823: --                      i => debug_data(i),
824: --                      t => debug_output_enable_n
825: --              );
826: --      end generate;
827: --
828: --
829: --      debug_pclk_iobuf : iobuf
830: --      generic map (
831: --              drive => 2,
832: --              iostandard => "lvcmos18",
833: --              slew => "slow")
834: --      port map (
835: --              o => ovm0_video_miso.pclk,
836: --              io => io_debug_ovm0_video_miso.pclk,
837: --              i => debug_pclk,
838: --              t => debug_output_enable_n
839: --      );
840: --
841: --      debug_href_iobuf : iobuf
842: --      generic map (
843: --              drive => 2,
844: --              iostandard => "lvcmos18",
845: --              slew => "slow")
846: --      port map (
847: --              o => ovm0_video_miso.href,
848: --              io => io_debug_ovm0_video_miso.href,
849: --              i => debug_href,
850: --              t => debug_output_enable_n
851: --      );
852: --
853: --      debug_vsync_iobuf : iobuf
854: --      generic map (
855: --              drive => 2,
856: --              iostandard => "lvcmos18",
857: --              slew => "slow")
858: --      port map (
859: --              o => ovm0_video_miso.vsync,
860: --              io => io_debug_ovm0_video_miso.vsync,
861: --              i => debug_vsync,
862: --              t => debug_output_enable_n
863: --      );
864:
865:
866:         ovm0_video_miso <= i_ovm0_video_miso;
867:         ovm1_video_miso <= i_ovm1_video_miso;
868:         ovm2_video_miso <= i_ovm2_video_miso;
869:         ovm3_video_miso <= i_ovm3_video_miso;
870:
871:
872:         process(clk_24m)
873:         begin
874:                 if ( rising_edge(clk_24m) ) then
875:                         mcu_intc_interrupt(0) <= i_switch1;
876:                 end if;
877:         end process;
878:
879:         mcu_intc_interrupt(7 downto 1) <= (others => '0');
880:
881:
882:         -- ============================================================================ --
883:         -- Error LEDs
884:         -- ============================================================================ --
885:
886:         o_error_led_n <= not error_led;
887:
888:         error_led0 <= mcu_gpo(GPIO_ERROR_LED)(0);
889:         mcu_gpi(GPIO_ERROR_LED) <= mcu_gpo(GPIO_ERROR_LED);
890:
891:         error_led1 <= flash_value;
892:         error_led2 <=
893:                 (not mctl_mport0_mosi.wr.empty) or
894:                 (not mctl_mport0_mosi.cmd.empty) or
895:                 (not mctl_mport2_mosi.rd.empty) or
896:                 (not mctl_mport2_mosi.cmd.empty);
897:         error_led3 <=
898:                         (mctl_mport0_mosi.wr.full) or
899:                         (mctl_mport0_mosi.cmd.full) or
900:                         (mctl_mport2_mosi.rd.full) or
```

```
901:                        (mctl_mport2_mosi.cmd.full);
902:
903:        error_led_src <= conv_integer(mcu_gpo(GPIO_ERROR_LED_SRC));
904:        mcu_gpi(GPIO_ERROR_LED_SRC) <= conv_std_logic_vector(error_led_src, 32);
905:
906:        with error_led_src select error_led <=
907:                        error_led0 when 0,
908:                        error_led1 when 1,
909:                        error_led2 when 2,
910:                        error_led3 when 3;
911:
912:
913:        leds_src <= mcu_gpo(GPIO_LEDS_SRC);
914:        mcu_gpi(GPIO_LEDS_SRC) <= mcu_gpo(GPIO_LEDS_SRC);
915:
916:        gen_led_mux :
917:        for i in 1 to 7 generate
918:        begin
919:                with leds_src(4*i+3 downto 4*i) select leds(i) <=
920:                        leds0(i) when x"0",
921:                        leds1(i) when x"1",
922:                        leds2(i) when x"2",
923:                        leds3(i) when x"3",
924:                        leds4(i) when x"4",
925:                        leds5(i) when x"5",
926:                        leds6(i) when x"6",
927:                        leds7(i) when x"7",
928:                        leds8(i) when x"8",
929:                        leds9(i) when x"9",
930:                        ledsA(i) when x"A",
931:                        ledsB(i) when x"B",
932:                        ledsC(i) when x"C",
933:                        ledsD(i) when x"D",
934:                        ledsE(i) when x"E",
935:                        ledsF(i) when x"F",
936:                        error_led when others;
937:        end generate;
938:
939:
940:        leds0 <= (others => '0');
941:
942:        leds1 <= mcu_gpo(GPIO_LEDS1)(7 downto 1);
943:        mcu_gpi(GPIO_LEDS1) <= mcu_gpo(GPIO_LEDS1);
944:
945:        leds2 <= mcu_gpo(GPIO_LEDS2)(7 downto 1);
946:        mcu_gpi(GPIO_LEDS2) <= mcu_gpo(GPIO_LEDS2);
947:
948:        leds3(2 downto 1) <= ram_status;
949:
950: --     leds4(7 downto 1) <= mctl_mport0_mosi.rd.data(7 downto 1);
951: --     leds5(7 downto 1) <= mctl_mport1_mosi.rd.data(7 downto 1);
952: --     leds6(7 downto 1) <= mctl_mport2_mosi.rd.data(7 downto 1);
953: --     leds7(7 downto 1) <= mctl_mport3_mosi.rd.data(7 downto 1);
954:
955:        leds9 <= (others => flash_value);
956:
957: --     ledsA(7 downto 1) <= ovm0_video_miso.data(7 downto 1);
958: --     ledsB(7 downto 1) <= i_ovm1_video_miso.data(7 downto 1);
959: --     ledsC(7 downto 1) <= i_ovm2_video_miso.data(7 downto 1);
960: --     ledsD(7 downto 1) <= i_ovm3_video_miso.data(7 downto 1);
961:
962:        ledsE <= (others => error_led);
963:        ledsF <= (others => '1');
964:
965:
966:
967:        flash_clk_div <= conv_integer(mcu_gpo(GPIO_FLASH_CLK_DIV));
968:        mcu_gpi(GPIO_FLASH_CLK_DIV) <= conv_std_logic_vector(flash_clk_div, 32);
969:
970:        flash_on <= conv_integer(mcu_gpo(GPIO_FLASH_ON));
971:        mcu_gpi(GPIO_FLASH_ON) <= conv_std_logic_vector(flash_on, 32);
972:
973:        flash_max <= conv_integer(mcu_gpo(GPIO_FLASH_MAX));
974:        mcu_gpi(GPIO_FLASH_MAX) <= conv_std_logic_vector(flash_max, 32);
975:
976:
977:        flash_clk_gate : cpt_clk_gate
978:        port map (
979:                i_clk => c3_clk0,
980:                i_enable => '1',
981:                i_div => flash_clk_div,
982:                o_clk_pgate => flash_clk_pgate,
983:                o_clk_ngate => open
984:        );
985:
986:        -- Generates a periodic pulse with a 32-bit programmable duty cycle
987:        flash_counter : cpt_upcounter
988:        generic map (
989:                INIT => 1
990:        )
991:        port map (
992:                i_clk => c3_clk0,
993:                i_enable => flash_clk_pgate,
994:                i_lowest => 0,
995:                i_highest => flash_max,
996:                i_increment => 1,
997:                i_clear => '0',
998:                i_preset => '0',
999:                o_count => flash_count,
1000:               o_carry => open
```

```
1001:         );
1002:
1003:         process(c3_clk0)
1004:         begin
1005:                 if ( rising_edge(c3_clk0) ) then
1006:                         if ( flash_count <= flash_on ) then
1007:                                 flash_value <= '1';
1008:                         else
1009:                                 flash_value <= '0';
1010:                         end if;
1011:                 end if;
1012:         end process;
1013:
1014:
1015:
1016:         o_ovm0_sccb_mosi <= ovm0_sccb_mosi;
1017:         o_ovm1_sccb_mosi <= ovm1_sccb_mosi;
1018:         o_ovm2_sccb_mosi <= ovm2_sccb_mosi;
1019:         o_ovm3_sccb_mosi <= ovm3_sccb_mosi;
1020:
1021:
1022:
1023:         uart_rxd <= i_mcu_uart_rx;
1024:         o_mcu_uart_tx <= uart_txd;
1025:
1026:
1027:         o_wifi_txd <= wifi_txd;
1028: --      wifi_rxd <= io_wifi_rxd;
1029: --      signal wifi_gpio0 : std_logic;
1030: --      signal wifi_gpio2 : std_logic;
1031:         o_wifi_ch_pd <= wifi_ch_pd;
1032:         o_wifi_rst <= wifi_rst;
1033:
1034:
1035:         incl_mcu:
1036:         if ( INCLUDE_MCU = "TRUE" ) generate
1037:
1038:                 mcu : cpt_mcu
1039:                 generic map (
1040:                         INCLUDE_IOBUS_MPORT => INCLUDE_IOBUS_MPORT,
1041:                         INCLUDE_SCCB => INCLUDE_SCCB
1042:                 )
1043:                 port map (
1044:                         i_clk => c3_clk0,
1045:                         --i_clk => clk_108,
1046:                         i_reset => reset_long,
1047:                         --o_debug_output_enable => debug_output_enable,
1048:                         --o_debug_src => debug_src,
1049:                         o_debug_output_enable => open,
1050:                         o_debug_src => open,
1051:                         -- mctl
1052:                         i_mctl_mport_mosi => mctl_mport3_mosi,
1053:                         o_mctl_mport_miso => mctl_mport3_miso,
1054:                         -- sccb
1055:                         io_ovm0_sccb_bidir => io_ovm0_sccb_bidir,
1056:                         o_ovm0_sccb_mosi => ovm0_sccb_mosi,
1057:                         io_ovm1_sccb_bidir => io_ovm1_sccb_bidir,
1058:                         o_ovm1_sccb_mosi => ovm1_sccb_mosi,
1059:                         io_ovm2_sccb_bidir => io_ovm2_sccb_bidir,
1060:                         o_ovm2_sccb_mosi => ovm2_sccb_mosi,
1061:                         io_ovm3_sccb_bidir => io_ovm3_sccb_bidir,
1062:                         o_ovm3_sccb_mosi => ovm3_sccb_mosi,
1063:                         -- uart
1064:                         i_uart_rx => uart_rxd,
1065:                         o_uart_tx => uart_txd,
1066:                         -- wifi
1067:                         o_wifi_txd => wifi_txd,
1068:                         io_wifi_rxd => io_wifi_rxd,
1069:                         o_wifi_rst => wifi_rst,
1070:                         io_wifi_gpio0 => io_wifi_gpio0,
1071:                         io_wifi_gpio2 => io_wifi_gpio2,
1072:                         o_wifi_ch_pd => wifi_ch_pd,
1073:                         -- external gpio
1074:                         i_gpi => mcu_gpi,
1075:                         o_gpo => mcu_gpo,
1076:                         -- interrupts
1077:                         i_intc_interrupt => mcu_intc_interrupt,
1078:                         o_intc_irq => mcu_intc_irq
1079:                 );
1080:
1081:         end generate;
1082:
1083:
1084:         ram_status(0) <= c3_error;
1085:         ram_status(1) <= c3_calib_done;
1086:
1087:         --c3_clk0 <= clk_108;
1088:         clk_108 <= c3_clk0;
1089:
1090:         incl_mctl:
1091:         if ( INCLUDE_MCTL = "TRUE" ) generate
1092:
1093:                 mctl_wrapper : cpt_mctl_wrapper
1094:                 generic map (
1095:                         INCLUDE_MCTL_CHIPSCOPE => INCLUDE_MCTL_CHIPSCOPE,
1096:                         C3_SIMULATION => C3_SIMULATION,
1097:                         C3_CALIB_SOFT_IP => C3_CALIB_SOFT_IP
1098:                 )
1099:                 port map (
1100:                         c3_sys_clk => clk_24m,
```

```
1101:                         c3_sys_rst_i => reset,
1102:                         ram_bidir => mctl_ram_bidir,
1103:                         ram_mosi => mctl_ram_mosi,
1104:                         c3_clk0 => c3_clk0,
1105:                         --c3_clk0 => open,
1106:                         c3_rst0 => open,
1107:                         c3_calib_done => c3_calib_done,
1108:                         mcb3_rzq => mcb3_rzq,
1109:                         --clk_108   => clk_108,
1110:                         --clk_108_n => clk_108_n,
1111:                         clk_108   => open,
1112:                         clk_108_n => open,
1113:                         mport0_miso => mctl_mport0_miso,
1114:                         mport0_mosi => mctl_mport0_mosi,
1115:                         mport1_miso => mctl_mport1_miso,
1116:                         mport1_mosi => mctl_mport1_mosi,
1117:                         mport2_miso => mctl_mport2_miso,
1118:                         mport2_mosi => mctl_mport2_mosi,
1119:                         mport3_miso => mctl_mport3_miso,
1120:                         mport3_mosi => mctl_mport3_mosi
1121:                 );
1122:
1123:
1124:                 incl_mctl_test :
1125:                 if ( INCLUDE_MCTL_TEST = "TRUE" ) generate
1126:
1127:                         -- Only connect port3 if MCU is disabled
1128:                         no_incl_mcu:
1129:                         if ( INCLUDE_MCU = "FALSE" ) generate
1130:                                 mctl_test_mport3_enabled <= '1';
1131:                                 mctl_test_mport3_mosi <= mctl_mport3_mosi;
1132:                                 mctl_mport3_miso <= mctl_test_mport3_miso;
1133:                         end generate no_incl_mcu; -- no_incl_mcu
1134:
1135:
1136:                         mctl_test_wrapper :  cpt_mctl_test_wrapper
1137:                         generic map (
1138:                                 C3_HW_TESTING => "FALSE"
1139:                         )
1140:                         port map        (
1141:                                 clk0 => c3_clk0,
1142:                                 --clk0 => clk_108,
1143:                                 rst0 => reset,
1144:                                 calib_done => c3_calib_done,
1145:                                 cmp_error => c3_cmp_error,
1146:                                 error => c3_error,
1147:                                 error_status => c3_error_status,
1148:                                 vio_modify_enable => c3_vio_modify_enable,
1149:                                 vio_data_mode_value => c3_vio_data_mode_value,
1150:                                 vio_addr_mode_value => c3_vio_addr_mode_value,
1151:                                 mport0_enabled => mctl_test_mport0_enabled,
1152:                                 mport0_miso => mctl_mport0_miso,
1153:                                 mport0_mosi => mctl_mport0_mosi,
1154:                                 mport1_enabled => mctl_test_mport1_enabled,
1155:                                 mport1_miso => mctl_mport1_miso,
1156:                                 mport1_mosi => mctl_mport1_mosi,
1157:                                 mport2_enabled => mctl_test_mport2_enabled,
1158:                                 mport2_miso => mctl_mport2_miso,
1159:                                 mport2_mosi => mctl_mport2_mosi,
1160:                                 mport3_enabled => mctl_test_mport3_enabled,
1161:                                 mport3_miso => mctl_test_mport3_miso,
1162:                                 mport3_mosi => mctl_test_mport3_mosi
1163:                         );
1164:
1165:                 end generate;
1166:
1167:         end generate;
1168:
1169:
1170:         mcu_gpi(GPIO_OVM_HREF)(0) <= i_ovm0_video_miso.href;
1171:         mcu_gpi(GPIO_OVM_HREF)(1) <= i_ovm1_video_miso.href;
1172:         mcu_gpi(GPIO_OVM_HREF)(2) <= i_ovm2_video_miso.href;
1173:         mcu_gpi(GPIO_OVM_HREF)(3) <= i_ovm3_video_miso.href;
1174:
1175:         mcu_gpi(GPIO_OVM_VSYNC)(0) <= ovm0_video_miso.vsync;
1176:         mcu_gpi(GPIO_OVM_VSYNC)(1) <= i_ovm1_video_miso.vsync;
1177:         mcu_gpi(GPIO_OVM_VSYNC)(2) <= i_ovm2_video_miso.vsync;
1178:         mcu_gpi(GPIO_OVM_VSYNC)(3) <= i_ovm3_video_miso.vsync;
1179:
1180:
1181:         ovm_mux_enable <= mcu_gpo(GPIO_OVM_MUX_ENABLE)(0);
1182:
1183:         ovm_mux_reset <= not ovm_mux_enable;
1184:
1185:         incl_cctl:
1186:         if ( INCLUDE_CCTL = "TRUE" ) generate
1187:
1188:                 ovm_mux: cpt_ovm_mux PORT MAP (
1189:                          i_clk => c3_clk0,
1190:                          i_reset => ovm_mux_reset,
1191:                          i0_frame_count => ovm0_bram_frame_number,
1192:                          i1_frame_count => ovm1_bram_frame_number,
1193:                          i2_frame_count => ovm2_bram_frame_number,
1194:                          i3_frame_count => ovm3_bram_frame_number,
1195:                          i_frame_addr0 => ovm_frame_addr0,
1196:                          i_frame_addr1 => ovm_frame_addr1,
1197:                          i_frame_addr2 => ovm_frame_addr2,
1198:                          i_frame_addr3 => ovm_frame_addr3,
1199:                          i0_line_offset => ovm0_line_offset,
1200:                          i1_line_offset => ovm1_line_offset,
```

```
1201:                           i2_line_offset => ovm2_line_offset,
1202:                           i3_line_offset => ovm3_line_offset,
1203:                           i0_words_read => ovm0_bram_words_read,
1204:                           i1_words_read => ovm1_bram_words_read,
1205:                           i2_words_read => ovm2_bram_words_read,
1206:                           i3_words_read => ovm3_bram_words_read,
1207:                           i0_line_count => ovm0_bram_line_number,
1208:                           i1_line_count => ovm1_bram_line_number,
1209:                           i2_line_count => ovm2_bram_line_number,
1210:                           i3_line_count => ovm3_bram_line_number,
1211:                           i0_rd_data => ovm0_bram_rd_data,
1212:                           i1_rd_data => ovm1_bram_rd_data,
1213:                           i2_rd_data => ovm2_bram_rd_data,
1214:                           i3_rd_data => ovm3_bram_rd_data,
1215:                           i0_burst_available => ovm0_bram_burst_available,
1216:                           i1_burst_available => ovm1_bram_burst_available,
1217:                           i2_burst_available => ovm2_bram_burst_available,
1218:                           i3_burst_available => ovm3_bram_burst_available,
1219:                           o0_rd_enable => ovm0_bram_rd_enable,
1220:                           o1_rd_enable => ovm1_bram_rd_enable,
1221:                           o2_rd_enable => ovm2_bram_rd_enable,
1222:                           o3_rd_enable => ovm3_bram_rd_enable,
1223:                           i_burst_length => burst_length,
1224:                           o_mport_miso => mctl_mport0_miso,
1225:                           i_mport_mosi => mctl_mport0_mosi
1226:                   );
1227:
1228:
1229:
1230:                 ovm_bram_enable <= mcu_gpo(GPIO_OVM_BRAM_ENABLE)(3 downto 0);
1231:
1232:                 ovm0_bram_reset <= not ovm_bram_enable(0);
1233:                 ovm1_bram_reset <= not ovm_bram_enable(1);
1234:                 ovm2_bram_reset <= not ovm_bram_enable(2);
1235:                 ovm3_bram_reset <= not ovm_bram_enable(3);
1236:
1237:
1238:                 ovm0_bram : cpt_ovm_bram PORT MAP (
1239:                         i_pclk => ovm0_video_miso.pclk,
1240:                         i_vsync => ovm0_video_miso.vsync,
1241:                         i_href => ovm0_video_miso.href,
1242:                         i_data => ovm0_video_miso.data,
1243:                         i_reset => ovm0_bram_reset,
1244:                         o_rd_data => ovm0_bram_rd_data,
1245:                         o_frame_number => ovm0_bram_frame_number,
1246:                         o_line_number => ovm0_bram_line_number,
1247:                         o_words_read => ovm0_bram_words_read,
1248:                         i_burst_length => burst_length,
1249:                         o_burst_available => ovm0_bram_burst_available,
1250:                         o_collision => o0_collision,
1251:                         i_clk => c3_clk0,
1252:                         i_rd_enable => ovm0_bram_rd_enable
1253:                 );
1254:
1255:                 ovm1_bram : cpt_ovm_bram PORT MAP (
1256:                         i_pclk => ovm1_video_miso.pclk,
1257:                         i_vsync => ovm1_video_miso.vsync,
1258:                         i_href => ovm1_video_miso.href,
1259:                         i_data => ovm1_video_miso.data,
1260:                         i_reset => ovm1_bram_reset,
1261:                         o_rd_data => ovm1_bram_rd_data,
1262:                         o_frame_number => ovm1_bram_frame_number,
1263:                         o_line_number => ovm1_bram_line_number,
1264:                         o_words_read => ovm1_bram_words_read,
1265:                         i_burst_length => burst_length,
1266:                         o_burst_available => ovm1_bram_burst_available,
1267:                         o_collision => o1_collision,
1268:                         i_clk => c3_clk0,
1269:                         i_rd_enable => ovm1_bram_rd_enable
1270:                 );
1271:
1272:                 ovm2_bram : cpt_ovm_bram PORT MAP (
1273:                         i_pclk => ovm2_video_miso.pclk,
1274:                         i_vsync => ovm2_video_miso.vsync,
1275:                         i_href => ovm2_video_miso.href,
1276:                         i_data => ovm2_video_miso.data,
1277:                         i_reset => ovm2_bram_reset,
1278:                         o_rd_data => ovm2_bram_rd_data,
1279:                         o_frame_number => ovm2_bram_frame_number,
1280:                         o_line_number => ovm2_bram_line_number,
1281:                         o_words_read => ovm2_bram_words_read,
1282:                         i_burst_length => burst_length,
1283:                         o_burst_available => ovm2_bram_burst_available,
1284:                         o_collision => o2_collision,
1285:                         i_clk => c3_clk0,
1286:                         i_rd_enable => ovm2_bram_rd_enable
1287:                 );
1288:
1289:                 ovm3_bram: cpt_ovm_bram PORT MAP (
1290:                         i_pclk => ovm3_video_miso.pclk,
1291:                         i_vsync => ovm3_video_miso.vsync,
1292:                         i_href => ovm3_video_miso.href,
1293:                         i_data => ovm3_video_miso.data,
1294:                         i_reset => ovm3_bram_reset,
1295:                         o_rd_data => ovm3_bram_rd_data,
1296:                         o_frame_number => ovm3_bram_frame_number,
1297:                         o_line_number => ovm3_bram_line_number,
1298:                         o_words_read => ovm3_bram_words_read,
1299:                         i_burst_length => burst_length,
1300:                         o_burst_available => ovm3_bram_burst_available,
```

```
1301:                         o_collision => o3_collision,
1302:                         i_clk => c3_clk0,
1303:                         i_rd_enable => ovm3_bram_rd_enable
1304:                 );
1305:
1306:         end generate incl_cctl;
1307:
1308:
1309:
1310:
1311:
1312:         incl_usb:
1313:         if ( INCLUDE_USB = "TRUE" ) generate
1314:
1315:                 usb : cpt_usb
1316:                 port map (
1317:                         i_usb_ctrl_miso => i_usb_ctrl_miso,
1318:                         o_usb_ctrl_mosi => o_usb_ctrl_mosi,
1319:                         io_usb_data_bidir => io_usb_data_bidir
1320:                 );
1321:
1322:         end generate incl_usb;
1323:
1324:         dummy_usb:
1325:         if ( INCLUDE_USB = "DUMMY" ) generate
1326:
1327:                 o_dummy_bank2_bottomright <= or_slv(dummy_usb_data_bidir.data);
1328:
1329:                 process(i_usb_ctrl_miso.clk)
1330:                 begin
1331:                         if ( rising_edge(i_usb_ctrl_miso.clk) ) then
1332:                                 dummy_usb_data_bidir.data <= io_usb_data_bidir.data;
1333:                         end if;
1334:                 end process;
1335:
1336:         end generate dummy_usb;
1337:
1338:
1339:
1340:
1341:
1342:         -- ========================================================================== --
1343:         -- VGA
1344:         -- ========================================================================== --
1345:
1346:         -- Select main output VGA source based on Microblaze flags
1347:         vga_src <= conv_integer(mcu_gpo(GPIO_VGA_SRC));
1348:
1349:         with vga_src select o_vga_mosi <=
1350:                 vga_mosi_test when 0,           -- If Microblaze flag VGA_SRC = 0
1351:                 vga_mosi_fixed when others;     -- If Microblaze flag VGA_SRC = 1
1352:
1353:
1354:         ------------------------
1355:         -- vga_fixed testcase --
1356:         ------------------------
1357:         vga_fixed_enable <= mcu_gpo(GPIO_VGA_FIXED_ENABLE)(0);
1358:
1359:         mctl_mport2_miso <= vga_mport_miso;
1360:         vga_mport_mosi <= mctl_mport2_mosi;
1361:
1362:         incl_vga_fixed:
1363:         if ( INCLUDE_VGA = "TEST" ) generate
1364:
1365: -- Commented out to avoid synth errors due to sharing vga_mport_miso signal between vga_fixed and vga_test2
1366: --              vga_fixed : cpt_vga_fixed
1367: --              port map (
1368: --                      i_clk => c3_clk0,
1369: --                      i_enable => vga_fixed_enable,
1370: --                      i_mport_mosi => vga_mport_mosi,
1371: --                      o_mport_miso => vga_mport_miso,
1372: --                      o_vga_mosi => vga_mosi_fixed
1373: --              );
1374:
1375:         end generate incl_vga_fixed;
1376:
1377:
1378:         ------------------------
1379:         -- vga_test testcase --
1380:         ------------------------
1381:         vga_test_enable <= mcu_gpo(GPIO_VGA_TEST_ENABLE)(0);
1382:         vga_test_mode <= mcu_gpo(GPIO_VGA_TEST_MODE)(0);
1383:
1384:         incl_vga_test:
1385:         if ( INCLUDE_VGA = "TEST" or INCLUDE_VGA = "TRUE" ) generate
1386:
1387:                 vga_test : cpt_vga_test
1388:                 port map (
1389:                         i_clk => clk_108,
1390:                         --i_clk => c3_clk0,
1391:                         i_enable => vga_test_enable,
1392:                         i_vga_test_mode => vga_test_mode,
1393:                         o_vga_mosi => vga_mosi_test
1394:                 );
1395:
1396:         end generate incl_vga_test;
1397:
1398:
1399:         ------------------------
1400:         -- vga_test2 testcase --
```

```
1401:         ------------------------
1402:
1403:
1404:
1405:         ovm_frame_addr0 <= mcu_gpo(GPIO_OVM_FRAME_ADDR0)(28 downto 0);
1406:         ovm_frame_addr1 <= mcu_gpo(GPIO_OVM_FRAME_ADDR1)(28 downto 0);
1407:         ovm_frame_addr2 <= mcu_gpo(GPIO_OVM_FRAME_ADDR2)(28 downto 0);
1408:         ovm_frame_addr3 <= mcu_gpo(GPIO_OVM_FRAME_ADDR3)(28 downto 0);
1409:
1410:
1411:         ovm0_line_offset <= conv_integer(mcu_gpo(GPIO_OVM0_LINE_OFFSET)(24 downto 0));
1412:         ovm1_line_offset <= conv_integer(mcu_gpo(GPIO_OVM1_LINE_OFFSET)(24 downto 0));
1413:         ovm2_line_offset <= conv_integer(mcu_gpo(GPIO_OVM2_LINE_OFFSET)(24 downto 0));
1414:         ovm3_line_offset <= conv_integer(mcu_gpo(GPIO_OVM3_LINE_OFFSET)(24 downto 0));
1415:
1416:
1417:
1418:         vga_frame_addr0 <= mcu_gpo(GPIO_VGA_FRAME_ADDR0)(28 downto 0);
1419:         vga_frame_addr1 <= mcu_gpo(GPIO_VGA_FRAME_ADDR1)(28 downto 0);
1420:         vga_frame_addr2 <= mcu_gpo(GPIO_VGA_FRAME_ADDR2)(28 downto 0);
1421:         vga_frame_addr3 <= mcu_gpo(GPIO_VGA_FRAME_ADDR3)(28 downto 0);
1422:
1423:
1424:         vga_mid_line_offset <= conv_integer(mcu_gpo(GPIO_VGA_MID_LINE_OFFSET)(25 downto 0));
1425:
1426:
1427:
1428:         vga_enable <= vga_fixed_enable ;
1429:
1430:         incl_vga_test2:
1431:         if ( INCLUDE_VGA = "TRUE" ) generate
1432:
1433:                 linebuf_test2 : cpt_linebuf
1434:                 port map (
1435:                         --i_clk => c3_clk0,
1436:                         i_clk => clk_108,
1437:                         i_enable => vga_enable,
1438:
1439:                         i_frame_addr0 => vga_frame_addr0,
1440:                         i_frame_addr1 => vga_frame_addr1,
1441:                         i_frame_addr2 => vga_frame_addr2,
1442:                         i_frame_addr3 => vga_frame_addr3,
1443:                         i_frame_number => frame_number,
1444:
1445:                         i_line_start => line_start,
1446:                         i_mid_line_offset => vga_mid_line_offset,
1447:                         i_line_number => line_number,
1448:
1449:                         i_burst_length => conv_std_logic_vector(15,6),  -- Max 6 bits
1450:                         i_pixel_number => pixel_number,
1451:
1452:                         i_mport_mosi => vga_mport_mosi,
1453:                         o_mport_miso => vga_mport_miso,
1454:
1455:                         o_linebuf_data => linebuf_data
1456:                 );
1457:
1458:                 vga_test2 : cpt_vga
1459:                 port map (
1460:                         --i_clk => c3_clk0,
1461:                         i_clk => clk_108,
1462:                         i_enable => vga_enable,
1463:
1464:                         i_linebuf_data => linebuf_data,
1465:                         o_line_start => line_start,
1466:
1467:                         o_pixel_number => pixel_number,
1468:                         o_line_number => line_number,
1469:                         o_frame_number => frame_number,
1470:
1471:                         o_vga_mosi => vga_mosi_fixed
1472:                 );
1473:
1474:         end generate incl_vga_test2;
1475:
1476:
1477:         -- ============================================================================ --
1478:         -- LED
1479:         -- ============================================================================ --
1480:
1481:         ledclk_div <= conv_integer(mcu_gpo(GPIO_LEDCLK_DIV));
1482:         mcu_gpi(GPIO_LEDCLK_DIV) <= mcu_gpo(GPIO_LEDCLK_DIV);
1483:
1484:         led_latch_div <= conv_integer(mcu_gpo(GPIO_LED_LATCH_DIV));
1485:         mcu_gpi(GPIO_LED_LATCH_DIV) <= mcu_gpo(GPIO_LED_LATCH_DIV);
1486:
1487:         leds_inst : cpt_leds
1488:         port map (
1489:                 --i_clk => clk_108,
1490:                 i_clk => c3_clk0,
1491:                 i_leds => leds,
1492:                 i_led_clk_div => ledclk_div,
1493:                 i_led_latch_div => led_latch_div,
1494:                 o_led_addr => o_led_addr
1495:         );
1496:
1497: end architecture;
```

```
 1:
 2: library ieee;
 3: use ieee.std_logic_1164.all;
 4:
 5: library unisim;
 6: use unisim.vcomponents.all;
 7:
 8: library usb;
 9: use usb.pkg_usb.all;
10:
11: entity cpt_usb is
12:
13:         port (
14:                 i_usb_ctrl_miso : in typ_usb_ctrl_miso;
15:                 o_usb_ctrl_mosi : out typ_usb_ctrl_mosi;
16:                 io_usb_data_bidir : inout typ_usb_data_bidir
17:         );
18:
19:
20: end cpt_usb;
21:
22: architecture Behavioral of cpt_usb is
23:
24:         signal usb_clk : std_logic := '0';
25:
26: begin
27:
28:
29:
30:         usb_clk_ibufg : ibufg
31:         port map (
32:                 i => i_usb_ctrl_miso.clk,
33:                 o => usb_clk
34:         );
35:
36:
37:
38:         o_usb_ctrl_mosi <= init_usb_ctrl_mosi;
39:         io_usb_data_bidir.data <= (others => 'X');
40:
41: end Behavioral;
42:
```

```
 1:
 2: library ieee;
 3: use ieee.std_logic_1164.all;
 4:
 5: package pkg_usb is
 6:
 7:         type typ_usb_ctrl_miso is record
 8:                 clk : std_logic;
 9:                 rxf_n : std_logic;
10:                 txe_n : std_logic;
11:         end record;
12:
13:         constant init_usb_ctrl_miso : typ_usb_ctrl_miso := (
14:                 clk => '1',
15:                 rxf_n => '1',
16:                 txe_n => '1'
17:         );
18:
19:
20:         type typ_usb_ctrl_mosi is record
21:                 rd_n : std_logic;
22:                 wr_n : std_logic;
23:                 oe_n : std_logic;
24:                 siwu_n : std_logic;
25:         end record;
26:
27:         constant init_usb_ctrl_mosi : typ_usb_ctrl_mosi := (
28:                 rd_n => '1',
29:                 wr_n => '1',
30:                 oe_n => '1',
31:                 siwu_n => '1'
32:         );
33:
34:
35:         type typ_usb_data_bidir is record
36:                 data : std_logic_vector(7 downto 0);
37:         end record;
38:
39:         constant init_usb_data_bidir : typ_usb_data_bidir := (
40:                 data => (others => '0')
41:         );
42:
43:
44:
45:         component cpt_usb is
46:                 port (
47:                         i_usb_ctrl_miso : in typ_usb_ctrl_miso;
48:                         o_usb_ctrl_mosi : out typ_usb_ctrl_mosi;
49:                         io_usb_data_bidir : inout typ_usb_data_bidir
50:                 );
51:         end component;
52:
53:
54: end pkg_usb;
55:
56: package body pkg_usb is
57:
58:
59: end pkg_usb;
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5:
  6: library unisim;
  7: use unisim.vcomponents.all;
  8:
  9: library util;
 10:
 11: -- Generate an oddr2 block for pins carrying output clocks
 12: -- The output clock may be divided by a factor of two
 13: -- If CLK_DIV2 = 0 : f_out = f_in
 14: -- If CLK_DIV2 > 0 : f_out = f_in / (2*CLK_DIV2)
 15:
 16: entity cpt_clkout is
 17:         port (
 18:                 i_enable : in std_logic;
 19:                 i_clk_div : in integer;
 20:                 i_clk : in std_logic;
 21:                 o_clk : out std_logic
 22:         );
 23:
 24: end cpt_clkout;
 25:
 26: architecture Behavioral of cpt_clkout is
 27:
 28:         signal clk : std_logic := '0';
 29:         signal clk_n : std_logic := '1';
 30:
 31:         signal clk_div : integer := 0;
 32:         signal div_count : integer := 0;
 33:         signal div_out : std_logic := '1';
 34:
 35:         signal reset : std_logic := '1';
 36:
 37: begin
 38:
 39:         clk <= i_clk;
 40:         clk_n <= not clk;
 41:
 42:
 43: --      gen_nodiv_clk :
 44: --      if ( CLK_DIV2 = 0 ) generate
 45: --
 46: --              clk_oddr2 : oddr2
 47: --              generic map(
 48: --                      DDR_ALIGNMENT => "NONE", -- Sets output alignment to "NONE", "C0", "C1"
 49: --                      INIT => '0', -- Sets initial state of the Q output to '0' or '1'
 50: --                      SRTYPE => "SYNC" -- Specifies "SYNC" or "ASYNC" set/reset
 51: --              )
 52: --              port map (
 53: --                      Q => o_clk, -- 1-bit output data to pin
 54: --                      C0 => clk, -- 1-bit clock input
 55: --                      C1 => clk_n, -- 1-bit clock input
 56: --                      CE => i_enable, -- 1-bit clock enable input
 57: --                      D0 => '0', -- 1-bit data input (associated with C0)
 58: --                      D1 => '1', -- 1-bit data input (associated with C1)
 59: --                      R => '0', -- 1-bit reset input
 60: --                      S => '0' -- 1-bit set input
 61: --              );
 62: --
 63: --      end generate gen_nodiv_clk;
 64: --
 65: --
 66: --
 67: --      gen_div_clk :
 68: --      if ( CLK_DIV2 /= 0 ) generate
 69:
 70:
 71:
 72:                 process(i_clk)
 73:                 begin
 74:                         --if ( rising_edge(i_clk) and i_enable = '1' ) then
 75:                         if ( rising_edge(i_clk) and (reset = '1' or i_enable = '1') ) then
 76:                                 reset <= '0';
 77:                                 if ( clk_div /= i_clk_div ) then
 78:                                         clk_div <= i_clk_div;
 79:                                         reset <= '1';
 80:                                 end if;
 81:                         end if;
 82:                 end process;
 83:
 84:
 85:
 86:
 87:
 88:
 89:                 process(i_clk, i_enable)
 90:                 begin
 91:                         if ( rising_edge(i_clk) and i_enable = '1' ) then
 92:                                 if ( reset = '1' ) then
 93:                                         div_count <= 1;
 94:                                 elsif ( div_count = clk_div ) then
 95:                                         div_count <= 1;
 96:                                         div_out <= not div_out;
 97:                                 else
 98:                                         div_count <= div_count + 1;
 99:                                 end if;
100:                         end if;
```

```
101:                end process;
102:
103:                clk_oddr2 : oddr2
104:                generic map(
105:                        DDR_ALIGNMENT => "NONE", -- Sets output alignment to "NONE", "C0", "C1"
106:                        INIT => '0', -- Sets initial state of the Q output to '0' or '1'
107:                        SRTYPE => "SYNC" -- Specifies "SYNC" or "ASYNC" set/reset
108:                )
109:                port map (
110:                        Q => o_clk, -- 1-bit output data to pin
111:                        C0 => clk, -- 1-bit clock input
112:                        C1 => clk_n, -- 1-bit clock input
113:                        CE => i_enable, -- 1-bit clock enable input
114:                        D0 => div_out, -- 1-bit data input (associated with C0)
115:                        D1 => div_out, -- 1-bit data input (associated with C1)
116:                        R => '0', -- 1-bit reset input
117:                        S => '0' -- 1-bit set input
118:                );
119: --
120: --      end generate gen_div_clk;
121:
122:
123:
124:
125:
126:
127: end Behavioral;
128:
```

```
 1:
 2: -- cpt_clk_gate.vhd
 3: --
 4: -- Clock gate generator
 5: -- Produces a periodic pulse that is high for one i_clk cycle
 6: -- with a frequency equal to i_clk divided by 2*i_div
 7:
 8: library ieee;
 9: use ieee.std_logic_1164.all;
10:
11: library util;
12: use util.pkg_util.all;
13:
14:
15: entity cpt_clk_gate is
16:         port (
17:                 i_clk : in std_logic;
18:                 i_enable : in std_logic;
19:                 i_div : in integer;
20:                 o_clk_pgate : out std_logic;
21:                 o_clk_ngate : out std_logic
22:         );
23: end cpt_clk_gate;
24:
25: architecture Behavioral of cpt_clk_gate is
26:
27:         signal count : integer := 0;
28:         signal carry : std_logic := '0';
29:         signal enable_n : std_logic := '1';
30:
31: begin
32:
33:         o_clk_pgate <= '1' when count = i_div and carry = '1' else '0';
34:         o_clk_ngate <= '1' when count = i_div and carry = '0' else '0';
35:
36:         enable_n <= not i_enable;
37:
38:         cycle_counter : cpt_upcounter
39:         generic map (
40:                 INIT => 1
41:         )
42:         port map (
43:                 i_clk => i_clk,
44:                 i_enable => i_enable,
45:                 i_lowest => 1,
46:                 i_highest => i_div,
47:                 i_increment => 1,
48:                 i_clear => enable_n,
49:                 i_preset => '0',
50:                 o_count => count,
51:                 o_carry => carry
52:         );
53:
54:
55: end Behavioral;
56:
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5: library unisim;
  6: use unisim.vcomponents.all;
  7:
  8:
  9: library util;
 10: use util.pkg_util.all;
 11:
 12:
 13: entity cpt_i2c is
 14:
 15:         port (
 16:
 17:                 i_clk : in std_logic;
 18:                 i_enable : in std_logic;
 19:
 20:                 i_scl_clk_div : in integer;
 21:
 22:                 i_addr : in std_logic_vector(6 downto 0);
 23:
 24:                 o_rd_data : out std_logic_vector(7 downto 0);
 25:                 o_rd_data_strobe : out std_logic;
 26:                 i_rd_start : in std_logic;
 27:                 o_rd_done : out std_logic;
 28:
 29:                 i_wr_data_available : in std_logic;
 30:                 i_wr_data : in std_logic_vector(7 downto 0);
 31:                 o_wr_data_strobe : out std_logic;
 32:                 i_wr_start : in std_logic;
 33:                 o_wr_done : out std_logic;
 34:
 35:                 io_i2c_scl : inout std_logic;
 36:                 io_i2c_sda : inout std_logic
 37:
 38:         );
 39:
 40: end cpt_i2c;
 41:
 42: architecture Behavioral of cpt_i2c is
 43:
 44:
 45:         signal i2c_clk_pgate : std_logic;
 46:
 47:         signal addr : std_logic_vector(6 downto 0);
 48:
 49:         signal sda_oe_n : std_logic;
 50:         signal sda_oe : std_logic;
 51:         signal sda_i : std_logic;
 52:         signal sda_o : std_logic := '1';
 53:
 54:         signal scl_oe_n : std_logic;
 55:         signal scl_oe : std_logic;
 56:         signal scl_i : std_logic;
 57:         signal scl_o : std_logic := '1';
 58:         signal scl_oddr : std_logic;
 59:
 60:
 61:
 62:         signal rd_active : std_logic;
 63:         signal wr_active : std_logic;
 64:
 65:         signal write_data : std_logic_vector(7 downto 0);
 66:         signal write_bit_count : integer;
 67:         signal write_word_count : integer;
 68:
 69:         signal read_data : std_logic_vector(7 downto 0);
 70:         signal read_bit_count : integer;
 71:         signal read_word_count : integer;
 72:
 73:         signal state : integer;
 74:
 75:         constant STATE_IDLE : integer := 16#00#;
 76:         constant STATE_START : integer := 16#10#;
 77:         constant STATE_WRITE : integer := 16#20#;
 78:         constant STATE_READ : integer := 16#30#;
 79:         constant STATE_WR_ACK : integer := 16#40#;
 80:         constant STATE_RD_ACK : integer := 16#50#;
 81:         constant STATE_STOP : integer := 16#60#;
 82:
 83:         signal wr_data_strobe : std_logic;
 84:         signal last_wr_data_strobe : std_logic;
 85:
 86:         signal rd_data_strobe : std_logic;
 87:         signal last_rd_data_strobe : std_logic;
 88:
 89:
 90:
 91: begin
 92:
 93:
 94:         sda_iobuf : iobuf
 95:         port map (
 96:                 io => io_i2c_sda,
 97:                 i => sda_o,
 98:                 o => sda_i,
 99:                 t => sda_oe_n
100:         );
```

```
101:
102:
103:
104:         scl_iobuf : iobuf
105:         port map (
106:                 io => io_i2c_scl,
107:                 i => scl_oddr,
108:                 o => scl_i,
109:                 t => scl_oe_n
110:         );
111:
112:
113: --      -- TODO: add clock streching support
114: --      scl_oddr2 : oddr2
115: --      port map (
116: --              Q => scl_oddr,
117: --              C0 => i_clk,
118: --              C1 => i_clk,
119: --              CE => i2c_clk_pgate,
120: --              D0 => scl_o,
121: --              D1 => scl_o,
122: --              R => '0',
123: --              S => '0'
124: --      );
125:
126:         scl_fd : fd
127:         port map (
128:                 Q => scl_oddr,
129:                 D => scl_o,
130:                 C => i_clk
131:         );
132:
133:
134:         i2c_clk_gate : cpt_clk_gate
135:         port map (
136:                 i_clk => i_clk,
137:                 i_enable => '1',
138:                 i_div => i_scl_clk_div,
139:                 o_clk_pgate => i2c_clk_pgate,
140:                 o_clk_ngate => open
141:         );
142:
143:
144:
145:
146:
147:         -- One-shot filters
148:         -- data strobes changes with i2c clock, but we need strobe width == period(i_clk)
149:
150:         o_wr_data_strobe <= wr_data_strobe and not last_wr_data_strobe;
151:
152:         process(i_clk)
153:         begin
154:                 if ( rising_edge(i_clk) ) then
155:                         last_wr_data_strobe <= wr_data_strobe;
156:                 end if;
157:         end process;
158:
159:
160:         o_rd_data_strobe <= rd_data_strobe and not last_rd_data_strobe;
161:
162:         process(i_clk)
163:         begin
164:                 if ( rising_edge(i_clk) ) then
165:                         last_rd_data_strobe <= rd_data_strobe;
166:                 end if;
167:         end process;
168:
169:
170:
171:         scl_oe_n <= not scl_oe;
172:         sda_oe_n <= not sda_oe;
173:
174:         process(i_clk)
175:         begin
176:                 if ( rising_edge(i_clk) ) then
177:
178:                         if ( i_enable = '0' ) then
179:                                 scl_o <= '1';
180:                                 scl_oe <= '1';
181:                                 sda_o <= '1';
182:                                 sda_oe <= '0';
183:                                 o_rd_done <= '0';
184:                                 o_wr_done <= '0';
185:                                 rd_data_strobe <= '0';
186:                                 wr_data_strobe <= '0';
187:                                 rd_active <= '0';
188:                                 wr_active <= '0';
189:                                 state <= STATE_IDLE;
190:
191:                         elsif ( i2c_clk_pgate = '1' ) then
192:
193:                                 case state is
194:                                         when STATE_IDLE+0 =>
195:                                                 scl_o <= '1';
196:                                                 scl_oe <= '1';
197:                                                 sda_o <= '1';
198:                                                 sda_oe <= '0';
199:                                                 o_rd_done <= '1';
200:                                                 o_wr_done <= '1';
```

```
201:                                            rd_data_strobe <= '0';
202:                                            wr_data_strobe <= '0';
203:                                            rd_active <= '0';
204:                                            wr_active <= '0';
205:                                            if ( i_rd_start = '1' ) then
206:                                                    addr <= i_addr;
207:                                                    read_word_count <= 1;
208:                                                    rd_active <= '1';
209:                                                    o_rd_done <= '0';
210:                                                    state <= STATE_START;
211:                                            end if;
212:                                            if ( i_wr_start = '1' ) then
213:                                                    addr <= i_addr;
214:                                                    wr_active <= '1';
215:                                                    o_wr_done <= '0';
216:                                                    state <= STATE_START;
217:                                            end if;
218:
219:                                    when STATE_START+0 | STATE_START+1 =>
220:                                            scl_o <= '1';
221:                                            scl_oe <= '1';
222:                                            sda_o <= '1';
223:                                            sda_oe <= '1';
224:                                            state <= state + 1;
225:
226:                                    when STATE_START+2 | STATE_START+3 =>
227:                                            scl_o <= '1';
228:                                            sda_o <= '0';
229:                                            state <= state + 1;
230:
231:                                    when STATE_START+4 =>
232:                                            scl_o <= '0';
233:                                            sda_o <= '0';
234:                                       write_data <= addr & rd_active;      -- LSb R/W bit: 0:write,1:read
235:                                            write_bit_count <= 7;
236:                                            state <= STATE_WRITE;
237:
238:
239:                                    when STATE_WRITE+0 =>
240:                                            wr_data_strobe <= '0';
241:                                            scl_o <= '0';
242:                                            sda_o <= write_data(write_bit_count);
243:                                            sda_oe <= '1';
244:                                            state <= state + 1;
245:
246:                                    when STATE_WRITE+1 | STATE_WRITE+2 =>
247:                                            scl_o <= '1';
248:                                            state <= state + 1;
249:
250:                                    when STATE_WRITE+3 =>
251:                                            scl_o <= '0';
252:                                            if ( write_bit_count = 0 ) then
253:                                                    state <= STATE_WR_ACK;
254:                                            else
255:                                                    write_bit_count <= write_bit_count - 1;
256:                                                    state <= STATE_WRITE;
257:                                            end if;
258:
259:
260:
261:                                    when STATE_WR_ACK+0 =>
262:                                            scl_o <= '0';
263:                                            sda_o <= '0';
264:                                            sda_oe <= '0';
265:                                            state <= state + 1;
266:
267:                                    when STATE_WR_ACK+1 | STATE_WR_ACK+2 =>
268:                                            scl_o <= '1';
269:                                            sda_o <= '0';
270:                                            state <= state + 1;
271:
272:                                    when STATE_WR_ACK+3 =>
273:                                            scl_o <= '0';
274:                                            state <= STATE_STOP; -- Default here if the following conditions are not met
275:                                            if ( rd_active = '1' and read_word_count /= 0 ) then
276:                                                    read_word_count <= read_word_count - 1;
277:                                                    read_bit_count <= 7;
278:                                                    state <= STATE_READ;
279:                                            end if;
280:                                            if ( wr_active = '1' and i_wr_data_available = '1' ) then
281:                                                    wr_data_strobe <= '1';
282:                                                    write_data <= i_wr_data;
283:                                                    write_bit_count <= 7;
284:                                                    state <= STATE_WRITE;
285:                                            end if;
286:
287:
288:
289:
290:
291:                                    when STATE_READ+0 =>
292:                                            --write_data_strobe <= '0';
293:                                            scl_o <= '0';
294:                                            sda_oe <= '0';
295:                                            state <= state + 1;
296:
297:                                    when STATE_READ+1 =>
298:                                            scl_o <= '1';
299:                                            state <= state + 1;
300:
```

```
301:                                        when STATE_READ+2 =>
302:                                                state <= state + 1;
303:
304:                                        when STATE_READ+3 =>
305:                                                scl_o <= '0';
306:                                                read_data(read_bit_count) <= sda_i;
307:                                                if ( read_bit_count = 0 ) then
308:                                                        state <= STATE_RD_ACK;
309:                                                else
310:                                                        read_bit_count <= read_bit_count - 1;
311:                                                        state <= STATE_READ;
312:                                                end if;
313:
314:
315:
316:
317:                                        when STATE_RD_ACK+0 =>
318:                                                scl_o <= '0';
319:                                                sda_o <= '1';
320:                                                sda_oe <= '1';
321:                                                rd_data_strobe <= '0';
322:                                                state <= state + 1;
323:
324:                                        when STATE_RD_ACK+1 | STATE_RD_ACK+2 =>
325:                                                scl_o <= '1';
326:                                                sda_o <= '1';
327:                                                state <= state + 1;
328:
329:                                        when STATE_RD_ACK+3 =>
330:                                                scl_o <= '0';
331:                                                o_rd_data <= read_data;
332:                                                rd_data_strobe <= '1';
333:                                                state <= STATE_STOP; -- Default here if the following conditions are not met
334:                                                if ( rd_active = '1' and read_word_count /= 0 ) then
335:                                                        read_word_count <= read_word_count - 1;
336:                                                        read_bit_count <= 8;
337:                                                        state <= STATE_READ;
338:                                                end if;
339:
340:
341:                                        when STATE_STOP+0 | STATE_STOP+1=>
342:                                                scl_o <= '0';
343:                                                sda_o <= '0';
344:                                                sda_oe <= '1';
345:                                                state <= state + 1;
346:
347:                                        when STATE_STOP+2 | STATE_STOP+3 =>
348:                                                scl_o <= '1';
349:                                                sda_o <= '0';
350:                                                state <= state + 1;
351:
352:                                        when STATE_STOP+4 | STATE_STOP+5 =>
353:                                                scl_o <= '1';
354:                                                sda_o <= '1';
355:                                                state <= state + 1;
356:
357:                                        when STATE_STOP+6 =>
358:                                                scl_o <= '1';
359:                                                sda_o <= '1';
360:                                                sda_oe <= '0';
361:                                                state <= STATE_IDLE;
362:
363:
364:
365:                                        when others =>
366:                                                state <= STATE_IDLE;
367:                                end case;
368:                        end if;
369:                end if;
370:        end process;
371:
372:
373:
374:
375:
376:
377:
378:
379:
380:
381:
382:
383:
384:
385:
386: end Behavioral;
387:
```

```
 1:
 2: -- util/cpt_upcounter.vhd
 3:
 4: -- Produces an integer o_count
 5: -- that increases from i_lowest to i_highest
 6: -- in steps of i_increment
 7: -- when i_enable is high
 8: -- during a rising edge of i_clk
 9:
10:
11: library ieee;
12: use ieee.std_logic_1164.all;
13:
14: --library util;
15: --use util.pkg_util.ALL;
16:
17:
18: entity cpt_upcounter is
19:         generic (
20:                 INIT : integer := -1
21:         );
22:         port (
23:                 i_clk : in std_logic;
24:                 i_enable : in std_logic;
25:                 i_lowest : in integer;
26:                 i_highest : in integer;
27:                 i_increment : in integer;
28:                 i_clear : in std_logic;
29:                 i_preset : in std_logic;
30:                 o_count : out integer := INIT;
31:                 o_carry : out std_logic
32:         );
33: end cpt_upcounter;
34:
35:
36: architecture Behavioral of cpt_upcounter is
37:
38: signal count : integer := INIT;
39: signal lowest : integer := 0;
40: signal highest : integer := 0;
41: signal increment : integer := 1;
42: signal reset : std_logic := '1';
43: signal carry : std_logic := '0';
44:
45: begin
46:
47:         o_count <= count;
48:         o_carry <= carry;
49:
50:
51:         process(i_clk)
52:         begin
53:                 --if ( rising_edge(i_clk) and i_enable = '1' ) then
54:                 if ( rising_edge(i_clk) and (reset = '1' or i_enable = '1') ) then
55:                         reset <= '0';
56:                         if ( lowest /= i_lowest ) then
57:                                 lowest <= i_lowest;
58:                                 reset <= '1';
59:                         end if;
60:                         if ( highest /= i_highest ) then
61:                                 highest <= i_highest;
62:                                 reset <= '1';
63:                         end if;
64:                         if ( increment /= i_increment ) then
65:                                 increment <= i_increment;
66:                                 reset <= '1';
67:                         end if;
68:                 end if;
69:         end process;
70:
71:
72:         process(i_clk, i_clear, i_preset)
73:         begin
74:                 if ( i_clear = '1' ) then
75:                         count <= lowest;
76:                         carry <= '0';
77:                 elsif ( i_preset = '1' ) then
78:                         count <= highest;
79:                         --carry <= '1';         -- omitted for now
80:                 elsif ( rising_edge(i_clk) and i_enable = '1' ) then
81:                         if ( reset = '1' ) then
82:                                 count <= lowest;
83:                                 carry <= '0';
84:                         elsif ( count >= highest ) then
85:                                 count <= lowest;
86:                                 carry <= not carry;
87:                         else
88:                                 count <= count + increment;
89:                         end if;
90:                 end if;
91:         end process;
92:
93: end Behavioral;
94:
95:
```

```
 1:
 2: library ieee;
 3: use ieee.std_logic_1164.all;
 4: use ieee.math_real.all;
 5:
 6: library util;
 7:
 8: package pkg_util is
 9:
10:         function cycles_f(duration, frequency : real) return integer is
11:         begin
12:                 return integer(ceil(duration * frequency));
13:         end cycles_f;
14:
15:         function cycles_p(duration, period : real) return integer is
16:         begin
17:                 return integer(ceil(duration / period));
18:         end cycles_p;
19:
20:
21:         function frequency(period : real) return real is
22:         begin
23:                 return 1.0 / period;
24:         end frequency;
25:
26:         function period(frequency : real) return real is
27:         begin
28:                 return 1.0 / frequency;
29:         end period;
30:
31:
32:
33:         component cpt_clkout is
34: --              generic (
35: --                      CLK_DIV2 : natural := 0
36: --              );
37:                 port (
38:                         i_enable : in std_logic;
39:                         i_clk_div : in integer;
40:                         i_clk : in std_logic;
41:                         o_clk : out std_logic
42:                 );
43:         end component;
44:
45:         component cpt_upcounter is
46:                 generic (
47:                         INIT : integer := -1
48:                 );
49:                 port (
50:                         i_clk : in std_logic;
51:                         i_enable : in std_logic;
52:                         i_lowest : in integer;
53:                         i_highest : in integer;
54:                         i_increment : in integer;
55:                         i_clear : in std_logic;
56:                         i_preset : in std_logic;
57:                         o_count : out integer := INIT;
58:                         o_carry : out std_logic
59:                 );
60:         end component;
61:
62:         component cpt_clk_gate is
63:                 port (
64:                         i_clk : in std_logic;
65:                         i_enable : in std_logic;
66:                         i_div : in integer;
67:                         o_clk_pgate : out std_logic;
68:                         o_clk_ngate : out std_logic
69:                 );
70:         end component;
71:
72:
73:
74:
75: end pkg_util;
76:
77:
78: package body pkg_util is
79: end pkg_util;
```

```
  1: ----------------------------------------------------------------------------------
  2: -- Author: Mark Mahony
  3: -- Designer: Chris Brown
  4: --
  5: -- Create Date:         12:47:03 07/07/2015
  6: -- Module Name:         cpt_linebuf
  7: -- Project Name:        ESE Capstone 2015
  8: -- Target Devices:      Xilinx Spartan-6
  9: -- Tool versions:       XISE 14.7
 10: -- Description:         Component to buffer data read from RAM.
 11: --
 12: -- Dependencies: Xilinx primitives, mctl and util libraries
 13: ----------------------------------------------------------------------------------
 14: library ieee;
 15: use ieee.std_logic_1164.all;
 16: use ieee.numeric_std.all;
 17:
 18: library unisim;
 19: use unisim.vcomponents.all;
 20:
 21: library mctl;
 22: use mctl.pkg_mctl.all;
 23:
 24: library util;
 25: use util.pkg_util.all;
 26:
 27:
 28: entity cpt_linebuf is
 29:         port (
 30:                 i_clk : in std_logic;
 31:                 i_enable : in std_logic;
 32:
 33:                 i_frame_addr0 : in std_logic_vector(28 downto 0);
 34:                 i_frame_addr1 : in std_logic_vector(28 downto 0);
 35:                 i_frame_addr2 : in std_logic_vector(28 downto 0);
 36:                 i_frame_addr3 : in std_logic_vector(28 downto 0);
 37:                 i_frame_number : in integer range 0 to 3 := 0;
 38:
 39:                 i_line_start : in std_logic;
 40:                 i_mid_line_offset : in integer range -(2**24) to (2**24)-1 := 0;
 41:                 i_line_number : in integer range -1024 to 1023 := -408;
 42:
 43:                 i_burst_length : in std_logic_vector(5 downto 0);
 44:                 i_pixel_number : in integer range -2048 to 2047;
 45:
 46:                 i_mport_mosi : in typ_mctl_mport_mosi;
 47:                 o_mport_miso : out typ_mctl_mport_miso;
 48:
 49:                 o_linebuf_data : out std_logic_vector(15 downto 0)
 50:         );
 51: end cpt_linebuf;
 52:
 53: architecture Behavioral of cpt_linebuf is
 54:
 55:         -- Linebuffer signals
 56:         signal linebuf_wr_addr : integer range 0 to (2**11)-1;
 57:         signal linebuf_wr_data : std_logic_vector (15 downto 0);
 58:         signal linebuf_wr_en : std_logic;
 59:
 60:         -- Frame signals
 61:         signal f_addr : std_logic_vector (28 downto 0);
 62:         --signal mid_line_offset : integer range -(2**29) to (2**29)-1;
 63:
 64:         -- MISO READ
 65:         signal mport_miso_rd_en : std_logic;
 66:         signal mport_miso_rd_en_d1 : std_logic;
 67:
 68:         -- MISO COMMAND
 69:         signal mport_miso_cmd_en : std_logic;
 70:         signal mport_miso_cmd_byte_addr : integer range 0 to (2**29)-1 := 0;
 71:
 72:         -- MOSI READ
 73:         signal mport_mosi_rd_empty : std_logic;
 74:         signal mport_mosi_rd_data : std_logic_vector (31 downto 0);
 75:
 76:         -- MOSI COMMAND
 77:         signal mport_mosi_cmd_full : std_logic;
 78:
 79:         signal words_requested : integer range 0 to (2**10)-1;
 80:         signal words_fetched : integer range 0 to (2**10)-1;
 81:
 82:
 83:
 84:         signal byte_addr_line_number : integer range 0 to 2**29-1;
 85:         signal byte_addr_f_addr : integer range 0 to 2**29-1;
 86:         signal byte_addr_mid_line_offset : integer range 0 to 2**29-1;
 87:         signal byte_addr_words_requested : integer range 0 to 2**29-1;
 88:         signal byte_addr_words_requested_offset : integer range 0 to 2**29-1;
 89:
 90:
 91: begin
 92:
 93: words_requested_counter : cpt_upcounter
 94:         port map (
 95:                 i_clk => i_clk,
 96:                 i_preset => '0',
 97:                 i_enable => mport_miso_cmd_en,
 98:                 i_clear => i_line_start,
 99:                 i_lowest => 0,
100:                 i_highest => 1023,
```

```
101:                i_increment => to_integer(unsigned(i_burst_length))+1,
102:                o_count => words_requested,
103:                o_carry => open
104:        );
105:
106: words_fetched_counter : cpt_upcounter
107:        port map (
108:                i_clk => i_clk,
109:                i_preset => '0',
110:                i_enable => mport_miso_rd_en,
111:                i_clear => i_line_start,
112:                i_lowest => 0,
113:                i_highest => 1023,
114:                i_increment => 1,
115:                o_count => words_fetched,
116:                o_carry => open
117:        );
118:
119: pixel_counter : cpt_upcounter
120:        port map (
121:                i_clk => i_clk,
122:                i_preset => '0',
123:                i_enable => linebuf_wr_en,
124:                i_clear => i_line_start,
125:                i_lowest => 0,
126:                i_highest => 2047,
127:                i_increment => 1,
128:                o_count => linebuf_wr_addr,
129:                o_carry => open
130:        );
131:
132: linebuf_upper_ramb8 : RAMB16_S9_S9
133:    generic map (
134:       INIT_A => X"000", --  Value of output RAM registers on Port A at startup
135:       INIT_B => X"000", --  Value of output RAM registers on Port B at startup
136:       SRVAL_A => X"000", --  Port A output value upon SSR assertion
137:       SRVAL_B => X"000", --  Port B output value upon SSR assertion
138:       WRITE_MODE_A => "WRITE_FIRST", --  WRITE_FIRST, READ_FIRST or NO_CHANGE
139:       WRITE_MODE_B => "WRITE_FIRST", --  WRITE_FIRST, READ_FIRST or NO_CHANGE
140:       SIM_COLLISION_CHECK => "ALL", -- "NONE", "WARNING", "GENERATE_X_ONLY", "ALL"
141:       -- The following INIT_xx declarations specify the initial contents of the RAM
142:       -- Address 0 to 511
143:       INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
144:       INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
145:       INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
146:       INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
147:       INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
148:       INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
149:       INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
150:       INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
151:       INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
152:       INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
153:       INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
154:       INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
155:       INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
156:       INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
157:       INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
158:       INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
159:       -- Address 512 to 1023
160:       INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
161:       INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
162:       INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
163:       INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
164:       INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
165:       INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
166:       INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
167:       INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
168:       INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
169:       INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
170:       INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
171:       INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
172:       INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
173:       INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
174:       INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
175:       INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
176:       -- Address 1024 to 1535
177:       INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
178:       INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
179:       INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
180:       INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
181:       INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
182:       INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
183:       INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
184:       INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
185:       INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
186:       INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
187:       INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
188:       INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
189:       INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
190:       INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
191:       INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
192:       INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
193:       -- Address 1536 to 2047
194:       INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
195:       INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
196:       INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
197:       INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
198:       INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
199:       INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
200:       INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
201:       INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
202:       INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
203:       INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
204:       INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
205:       INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
206:       INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
207:       INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
208:       INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
209:       INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",
210:       -- The next set of INITP_xx are for the parity bits
211:       -- Address 0 to 511
212:       INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
213:       INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
214:       -- Address 512 to 1023
215:       INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
216:       INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
217:       -- Address 1024 to 1535
218:       INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
219:       INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
220:       -- Address 1536 to 2047
221:       INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
222:       INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000")
223:    port map (
224:       DOA => open,      -- Port A 8-bit Data Output
225:       DOB => o_linebuf_data(15 downto 8),      -- Port B 8-bit Data Output
226:       DOPA => open,    -- Port A 1-bit Parity Output
227:       DOPB => open,    -- Port B 1-bit Parity Output
228:       ADDRA => std_logic_vector(to_unsigned(linebuf_wr_addr mod 2048, 11)),  -- Port A 11-bit Address Input
229:       ADDRB => std_logic_vector(to_unsigned(i_pixel_number mod 2048, 11)),  -- Port B 11-bit Address Input
230:       CLKA => i_clk,    -- Port A Clock
231:       CLKB => i_clk,    -- Port B Clock
232:       DIA => linebuf_wr_data(15 downto 8),      -- Port A 8-bit Data Input
233:       DIB => (others => '0'),      -- Port B 8-bit Data Input
234:       DIPA => (others => '0'),    -- Port A 1-bit parity Input
235:       DIPB => (others => '0'),    -- Port-B 1-bit parity Input
236:       ENA => '1',      -- Port A RAM Enable Input
237:       ENB => '1',      -- PortB RAM Enable Input
238:       SSRA => '0',    -- Port A Synchronous Set/Reset Input
239:       SSRB => '0',    -- Port B Synchronous Set/Reset Input
240:       WEA => linebuf_wr_en,      -- Port A Write Enable Input
241:       WEB => '0'       -- Port B Write Enable Input
242:    );
243:
244: linebuf_lower_ramb8 : RAMB16_S9_S9
245:    generic map (
246:       INIT_A => X"000", --  Value of output RAM registers on Port A at startup
247:       INIT_B => X"000", --  Value of output RAM registers on Port B at startup
248:       SRVAL_A => X"000", --  Port A output value upon SSR assertion
249:       SRVAL_B => X"000", --  Port B output value upon SSR assertion
250:       WRITE_MODE_A => "WRITE_FIRST", --  WRITE_FIRST, READ_FIRST or NO_CHANGE
251:       WRITE_MODE_B => "WRITE_FIRST", --  WRITE_FIRST, READ_FIRST or NO_CHANGE
252:       SIM_COLLISION_CHECK => "ALL", -- "NONE", "WARNING", "GENERATE_X_ONLY", "ALL"
253:       -- The following INIT_xx declarations specify the initial contents of the RAM
254:       -- Address 0 to 511
255:       INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
256:       INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
257:       INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
258:       INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
259:       INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
260:       INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
261:       INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
262:       INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
263:       INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
264:       INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
265:       INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
266:       INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
267:       INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
268:       INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
269:       INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
270:       INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
271:       -- Address 512 to 1023
272:       INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
273:       INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
274:       INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
275:       INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
276:       INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
277:       INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
278:       INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
279:       INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
280:       INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
281:       INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
282:       INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
283:       INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
284:       INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
285:       INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
286:       INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
287:       INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
288:       -- Address 1024 to 1535
289:       INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
290:       INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
291:       INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
292:       INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
293:       INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
294:       INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
295:       INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
296:       INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
297:       INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
298:       INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
299:       INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
300:       INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
301:       INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
302:       INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
303:       INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
304:       INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
305:       -- Address 1536 to 2047
306:       INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
307:       INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
308:       INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
309:       INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
310:       INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
311:       INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
312:       INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
313:       INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
314:       INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
315:       INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
316:       INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
317:       INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
318:       INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
319:       INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
320:       INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
321:       INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",
322:       -- The next set of INITP_xx are for the parity bits
323:       -- Address 0 to 511
324:       INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
325:       INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
326:       -- Address 512 to 1023
327:       INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
328:       INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
329:       -- Address 1024 to 1535
330:       INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
331:       INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
332:       -- Address 1536 to 2047
333:       INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
334:       INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000")
335:    port map (
336:       DOA => open,      -- Port A 8-bit Data Output
337:       DOB => o_linebuf_data(7 downto 0),      -- Port B 8-bit Data Output
338:       DOPA => open,    -- Port A 1-bit Parity Output
339:       DOPB => open,    -- Port B 1-bit Parity Output
340:       ADDRA => std_logic_vector(to_unsigned(linebuf_wr_addr mod 2048, 11)),  -- Port A 11-bit Address Input
341:       ADDRB => std_logic_vector(to_unsigned(i_pixel_number mod 2048, 11)),  -- Port B 11-bit Address Input
342:       CLKA => i_clk,    -- Port A Clock
343:       CLKB => i_clk,    -- Port B Clock
344:       DIA => linebuf_wr_data(7 downto 0),      -- Port A 8-bit Data Input
345:       DIB => (others => '0'),      -- Port B 8-bit Data Input
346:       DIPA => (others => '0'),    -- Port A 1-bit parity Input
347:       DIPB => (others => '0'),    -- Port-B 1-bit parity Input
348:       ENA => '1',      -- Port A RAM Enable Input
349:       ENB => '1',      -- PortB RAM Enable Input
350:       SSRA => '0',    -- Port A Synchronous Set/Reset Input
351:       SSRB => '0',    -- Port B Synchronous Set/Reset Input
352:       WEA => linebuf_wr_en,      -- Port A Write Enable Input
353:       WEB => '0'       -- Port B Write Enable Input
354:    );
355:
356: inst : FD
357:         port map (
358:                 D => mport_miso_rd_en,
359:                 C => i_clk,
360:                 Q => mport_miso_rd_en_d1
361:         );
362:
363:         -- Input & Output
364:         o_mport_miso.wr.clk <= i_clk;
365:         o_mport_miso.wr.en <= '0';
366:         o_mport_miso.wr.mask <= "1111";
367:         o_mport_miso.wr.data <= (others => '0');
368:
369:         o_mport_miso.cmd.instr <= "011";        -- Read mode with auto-precharge
370:         o_mport_miso.cmd.clk <= i_clk;
371:         o_mport_miso.cmd.en <= mport_miso_cmd_en;
372:         o_mport_miso.cmd.bl <= i_burst_length;
373:
374:         o_mport_miso.rd.clk <= i_clk;
375:         o_mport_miso.rd.en <= mport_miso_rd_en;
376:
377:         mport_mosi_cmd_full <= i_mport_mosi.cmd.full;
378:         mport_mosi_rd_empty <= i_mport_mosi.rd.empty;
379:         mport_mosi_rd_data <= i_mport_mosi.rd.data;
380:
381:
382:
383:         byte_addr_line_number <= (i_line_number * 2048);
384:         byte_addr_f_addr <= to_integer(unsigned(f_addr));
385:         byte_addr_mid_line_offset <= (i_mid_line_offset * 2048);
386:         byte_addr_words_requested <= (words_requested * 4);
387:         byte_addr_words_requested_offset <= ((words_requested-320) * 4);
388:
389:
390:
391:         -- Address generation
392:         mport_miso_cmd_byte_addr <=
393:                 (
394:                         byte_addr_line_number +
395:                         byte_addr_f_addr +
396:                         byte_addr_mid_line_offset +
397:                         byte_addr_words_requested_offset
398:                 ) mod (2**29)
399:         when (words_requested >= 320)   else
400:                 (
```

```
401:                        byte_addr_line_number +
402:                        byte_addr_f_addr +
403:                        byte_addr_words_requested
404:                ) mod (2**29);
405:
406:        o_mport_miso.cmd.byte_addr <= std_logic_vector(to_unsigned(mport_miso_cmd_byte_addr, o_mport_miso.cmd.byte_addr'length));
407:        --mid_line_offset <= (i_mid_line_offset) when (words_requested >= 320) else 0;
408:
409:        -- Counter enable
410:        mport_miso_cmd_en <= '1' when (i_line_start = '0') and
411:                                              (mport_mosi_cmd_full = '0') and
412:                                              (words_requested < 640) and
413:                                              (64-((words_requested - words_fetched)) > to_integer(unsigned(i_burst_length))+1)
414:                                              --((words_requested - words_fetched) <= to_integer(unsigned(i_burst_length))+1)
415:                                else '0';
416:
417:        -- Data path
418:        mport_miso_rd_en <= (not mport_mosi_rd_empty) and (not mport_miso_rd_en_d1);
419:        linebuf_wr_en <= mport_miso_rd_en or mport_miso_rd_en_d1;
420:        linebuf_wr_data <= mport_mosi_rd_data(15 downto 0) when (mport_miso_rd_en_d1 = '1') else mport_mosi_rd_data(31 downto 16);
421:
422:
423:        process (i_clk)
424:        begin
425:                if rising_edge(i_clk) then
426:                        -- Get new frame offset address when there is a new line
427:                        if ( i_line_start = '1' ) then
428:                                case i_frame_number is
429:                                        when 0 =>
430:                                                f_addr <= i_frame_addr0;
431:                                        when 1 =>
432:                                                f_addr <= i_frame_addr1;
433:                                        when 2 =>
434:                                                f_addr <= i_frame_addr2;
435:                                        when 3 =>
436:                                                f_addr <= i_frame_addr3;
437:                                end case;
438:                        end if;
439:                end if;
440:        end process;
441:
442: end Behavioral;
443:
```

```
--
-- VGA component
--
-- Splits RGB data vector into separate RGB channels
-- Generates hsync and vsync
-- Keeps track of pixel count, line count, frame count and when new line starts
--

library ieee;
use ieee.std_logic_1164.all;

library unisim;
use unisim.vcomponents.all;

library vga;
use vga.pkg_vga.all;

library util;
use util.pkg_util.all;


entity cpt_vga is
	port (
		i_clk : in std_logic;
		i_enable : in std_logic;

		i_linebuf_data : in std_logic_vector(15 downto 0);	-- Input RGB data
		o_line_start : out std_logic;	-- Beginning of line indicator flag

		o_pixel_number : out integer range -2048 to 2047;
		o_line_number : out integer range -1024 to 1023;
		o_frame_number : out integer range 0 to 3;

		o_vga_mosi : out typ_vga_mosi := init_vga_mosi  -- Output RGB data, hsync, vsync
	);
end cpt_vga;

architecture Behavioral of cpt_vga is

	signal clear_count : std_logic := '0';
	signal increment_pixel : std_logic := '0';
	signal pixel_number : integer range -2048 to 2047;
	signal line_number : integer range -1024 to 1023;
	signal h_blank : std_logic := '0';
	signal h_blank_d1 : std_logic := '0';
	signal v_blank : std_logic := '0';
	signal v_blank_d1 : std_logic := '0';
	signal h_sync : std_logic := '0';
	signal h_sync_d1 : std_logic := '0';
	signal v_sync : std_logic := '0';
	signal v_sync_d1 : std_logic := '0';
	signal increment_line : std_logic := '0';
	signal increment_frame : std_logic := '0';

begin

	--
	-- Counters
	--

	-- Count pixels from -408 to 1279
	-- Active pixel starts at 0
	pixel_counter: cpt_upcounter
	port map (
		i_clk => i_clk,
		i_enable => increment_pixel,
		i_lowest => (-H_BP - H_PULSE - H_FP),
		i_highest => PIXELS_PER_LINE-1,
		i_increment => 1,
		i_clear => '0',
		i_preset => clear_count,
		o_count => pixel_number,
		o_carry => open
	);

	-- Count lines from -42 to 1023
	-- Active line starts at 0
	line_counter: cpt_upcounter
	port map (
		i_clk => i_clk,
		i_enable => increment_line,
		i_lowest => (-V_BP - V_PULSE - V_FP),
		i_highest => MAX_LINES-1,
		i_increment => 1,
		i_clear => '0',
		i_preset => clear_count,
		o_count => line_number,
		o_carry => open
	);

	-- Count frames from 0 to 3
	frame_counter: cpt_upcounter
	port map (
		i_clk => i_clk,
		i_enable => increment_frame,
		i_lowest => 0,
		i_highest => 3,
		i_increment => 1,
		i_clear => '0',
		i_preset => clear_count,
```

```
101:                 o_count => o_frame_number,
102:                 o_carry => open
103:         );
104:
105:
106:         --
107:         -- Data latches
108:         --
109:
110:         -- Sync increment_pixel with clk
111:         increment_pixel_fd: FD
112:         port map (
113:                 D => i_enable,
114:                 C => i_clk,
115:                 Q => increment_pixel
116:         );
117:
118: --      -- Sync h_blank with clk
119: --      h_blank_d1_fd: FD
120: --      port map (
121: --              D => h_blank,
122: --              C => i_clk,
123: --              Q => h_blank_d1
124: --      );
125:
126:         -- Sync v_blank with clk
127:         v_blank_d1_fd: FD
128:         port map (
129:                 D => v_blank,
130:                 C => i_clk,
131:                 Q => v_blank_d1
132:         );
133:
134:         -- Sync hsync with clk
135:         h_sync_d1_fd: FD
136:         port map (
137:                 D => h_sync,
138:                 C => i_clk,
139:                 Q => h_sync_d1
140:         );
141:
142:         -- Sync vsync with clk
143:         v_sync_d1_fd: FD
144:         port map (
145:                 D => v_sync,
146:                 C => i_clk,
147:                 Q => v_sync_d1
148:         );
149:
150:         -- Sync line_start with clk
151:         line_start_fd: FD
152:         port map (
153:                 D => increment_line,
154:                 C => i_clk,
155:                 Q => o_line_start
156:         );
157:
158:         -- Sync hsync with clk
159:         vga_hsync_fd: FD
160:         port map (
161:                 D => h_sync_d1,
162:                 C => i_clk,
163:                 Q => o_vga_mosi.hsync
164:         );
165:
166:         -- Sync vsync with clk
167:         vga_vsync_fd: FD
168:         port map (
169:                 D => v_sync_d1,
170:                 C => i_clk,
171:                 Q => o_vga_mosi.vsync
172:         );
173:
174:
175:         clear_count <= not i_enable;    -- Reset all counters when not enabled
176:
177:         o_pixel_number <= pixel_number;
178:         o_line_number <= line_number;
179:
180:
181: --      h_blank <= '1'
182: --              when pixel_number < 0 -- Horizontal blanking period
183: --              else '0';
184:
185:         process (i_clk)
186:         begin
187:                 if rising_edge(i_clk) then
188:                         -- Horizontal blanking period
189:                         if ( pixel_number < 0 ) then
190:                                 h_blank_d1 <= '1';
191:                         else
192:                                 h_blank_d1 <= '0';
193:                         end if;
194:                 end if;
195:         end process;
196:
197:
198:         v_blank <= '1'
199:                 when line_number < 0 -- Vertical blanking period
200:                 else '0';
```

```
201:
202:         h_sync <= '1'
203:                 when pixel_number < -H_BP and pixel_number >= (-H_BP - H_PULSE) -- Horizontal sync pulse time
204:                 else '0';
205:
206:         v_sync <= '1'
207:                 when line_number < -V_BP and line_number >= (-V_BP - V_PULSE) -- Vertical sync pulse time
208:                 else '0';
209:
210:         increment_line <= '1'
211:                 when pixel_number = PIXELS_PER_LINE-1 and increment_pixel = '1' -- Reached end of current line
212:                 else '0';
213:
214:         increment_frame <= '1'
215:                 when pixel_number = PIXELS_PER_LINE-1 and line_number = MAX_LINES-1 and increment_pixel = '1' -- Reached last line in frame
216:                 else '0';
217:
218:         -- Mux and split RGB data
219:         process (i_clk)
220:         begin
221:                 if rising_edge(i_clk) then
222:                         case ( h_blank_d1 or v_blank_d1 ) is
223:                                 when '0' =>
224:                                         -- Only use 4 bits of colour channel data, throw away LSb
225:                                         -- RGB565 colour format uses 5 bits for red and blue, 6 bits for green
226:                                         -- http://www.theimagingsource.com/en_US/support/documentation/icimagingcontrol-class/PixelformatRGB565.htm
227:                                         o_vga_mosi.red <= i_linebuf_data(15 downto 12);
228:                                         o_vga_mosi.green <= i_linebuf_data(10 downto 7);
229:                                         o_vga_mosi.blue <= i_linebuf_data(4 downto 1);
230:
231:                                 when others =>
232:                                         -- If either horizontal or vertical blanking periods active,
233:                                         -- there is no colour data output
234:                                         o_vga_mosi.red <= (others => '0');
235:                                         o_vga_mosi.green <= (others => '0');
236:                                         o_vga_mosi.blue <= (others => '0');
237:
238:                         end case;
239:                 end if;
240:         end process;
241:
242: end Behavioral;
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4: use ieee.numeric_std.all;
  5: --use ieee.std_logic_arith.all;
  6: --use ieee.std_logic_unsigned.all;
  7:
  8:
  9:
 10: library mctl;
 11: use mctl.pkg_mctl.all;
 12:
 13: library vga;
 14: use vga.pkg_vga.all;
 15:
 16: library util;
 17: use util.pkg_util.all;
 18:
 19:
 20: entity cpt_vga_fixed is
 21:
 22:         generic (
 23:                 SMALL_FRAME : string := "FALSE"
 24:         );
 25:
 26:                 port  (
 27:                         i_clk : in std_logic;
 28:                         i_enable : in std_logic;
 29:                         i_mport_mosi : in typ_mctl_mport_mosi;
 30:                         o_mport_miso : out typ_mctl_mport_miso;
 31:                         o_vga_mosi : out typ_vga_mosi
 32:                 );
 33:
 34: end cpt_vga_fixed;
 35:
 36:
 37: architecture behavioral of cpt_vga_fixed is
 38:
 39:         function f(v1:integer; v2:integer) return integer is
 40:                 begin
 41:                 if (SMALL_FRAME = "FALSE") then
 42:                   return v1;
 43:                 else
 44:                   return v2;
 45:                 end if;
 46:    end function;
 47:
 48:
 49:         -- ----------------------------------
 50:         -- Characteristic timing constants
 51:         -- ----------------------------------
 52:
 53:         constant H_SUBFRAME_WIDTH : integer := 640;
 54:         constant V_SUBFRAME_WIDTH : integer := 480;
 55:
 56:         constant H_ACTIVE_WIDTH : integer := f(1280, 12);
 57:         constant H_FRONTPORCH_WIDTH : integer := f(48, 2);
 58:         constant H_SYNC_WIDTH : integer := f(112, 1);
 59:         constant H_BACKPORCH_WIDTH : integer := f(248, 2);
 60:
 61:         constant V_ACTIVE_WIDTH : integer := f(1024, 12);
 62:         constant V_FRONTPORCH_WIDTH : integer := f(1, 1);
 63:         constant V_SYNC_WIDTH : integer := f(3, 1);
 64:         constant V_BACKPORCH_WIDTH : integer := f(38, 2);
 65:
 66:
 67:         -- ------------------------------------------------------
 68:         -- Derived timing constants (do not modify without reason)
 69:         -- ------------------------------------------------------
 70:
 71:         constant H_ACTIVE_FIRST : integer := 0;
 72:         constant H_ACTIVE_LAST : integer := H_ACTIVE_FIRST + H_ACTIVE_WIDTH - 1;
 73:
 74:         constant H_SUBFRAME_FIRST : integer := 0;
 75:         constant H_SUBFRAME_LAST : integer := H_SUBFRAME_FIRST + H_SUBFRAME_WIDTH - 1;
 76:
 77:         constant H_FRONTPORCH_FIRST : integer := H_ACTIVE_LAST + 1;
 78:         constant H_FRONTPORCH_LAST : integer := H_FRONTPORCH_FIRST + H_FRONTPORCH_WIDTH - 1;
 79:
 80:         constant H_SYNC_FIRST : integer := H_FRONTPORCH_LAST + 1;
 81:         constant H_SYNC_LAST : integer := H_SYNC_FIRST + H_SYNC_WIDTH - 1;
 82:
 83:         constant H_BACKPORCH_FIRST : integer := H_SYNC_LAST + 1;
 84:         constant H_BACKPORCH_LAST : integer := H_BACKPORCH_FIRST + H_BACKPORCH_WIDTH - 1;
 85:
 86:         constant H_BLANK_FIRST : integer := H_FRONTPORCH_FIRST;
 87:         constant H_BLANK_LAST : integer := H_BACKPORCH_LAST;
 88:
 89:         constant H_FRAME_FIRST : integer := H_ACTIVE_FIRST;
 90:         constant H_FRAME_LAST : integer := H_BACKPORCH_LAST;
 91:
 92:         constant V_ACTIVE_FIRST : integer := 0;
 93:         constant V_ACTIVE_LAST : integer := V_ACTIVE_FIRST + V_ACTIVE_WIDTH - 1;
 94:
 95:         constant V_SUBFRAME_FIRST : integer := 0;
 96:         constant V_SUBFRAME_LAST : integer := V_SUBFRAME_FIRST + V_SUBFRAME_WIDTH - 1;
 97:
 98:         constant V_FRONTPORCH_FIRST : integer := V_ACTIVE_LAST + 1;
 99:         constant V_FRONTPORCH_LAST : integer := V_FRONTPORCH_FIRST + V_FRONTPORCH_WIDTH - 1;
100:
```

```
101:         constant V_SYNC_FIRST : integer := V_FRONTPORCH_LAST + 1;
102:         constant V_SYNC_LAST : integer := V_SYNC_FIRST + V_SYNC_WIDTH - 1;
103:
104:         constant V_BACKPORCH_FIRST : integer := V_SYNC_LAST + 1;
105:         constant V_BACKPORCH_LAST : integer := V_BACKPORCH_FIRST + V_BACKPORCH_WIDTH - 1;
106:
107:         constant V_BLANK_FIRST : integer := V_FRONTPORCH_FIRST;
108:         constant V_BLANK_LAST : integer := V_BACKPORCH_LAST;
109:
110:         constant V_FRAME_FIRST : integer := V_ACTIVE_FIRST;
111:         constant V_FRAME_LAST : integer := V_BACKPORCH_LAST;
112:
113:
114:         -- Video timing generation
115:         signal h_count : integer := 0;
116:         signal h_active : std_logic := '1';
117:         signal h_subframe_active : std_logic := '1';
118:         signal h_sync : std_logic := '0';
119:
120:
121:
122:
123:
124:         constant LINE_BURST_LENGTH : integer := 8;
125:
126:
127:
128:         signal h_fetch_count : integer;
129:
130:
131:         signal v_counter_enable : std_logic := '1';
132:         signal v_count : integer := 0;
133:         signal v_active : std_logic := '1';
134:         signal v_subframe_active : std_logic := '1';
135:         signal v_sync : std_logic := '0';
136:
137:         signal frame_counter_enable : std_logic := '1';
138:         signal frame_count : integer := 0;
139:         signal frame_active : std_logic := '1';
140:
141:         signal reset : std_logic := '0';
142:
143:
144:         signal addr_latch : std_logic_vector(25 downto 0) := (others => '0');
145:
146:         signal cmd_empty_d1 : std_logic;
147:
148:
149:         signal red : integer := 0;
150:         signal green : integer := 0;
151:         signal blue : integer := 0;
152:
153:
154:         signal red_vector : std_logic_vector(15 downto 0) := (others => '0');
155:         signal green_vector : std_logic_vector(15 downto 0) := (others => '0');
156:         signal blue_vector : std_logic_vector(15 downto 0) := (others => '0');
157:
158:
159:         signal frame_vector : std_logic_vector(15 downto 0) := (others => '0');
160:
161: begin
162:
163:
164:         reset <= not i_enable;
165:
166:
167: --      red_vector <= std_logic_vector(to_unsigned(4*frame_count - 3*h_count - 5*v_count, 16));
168: --      green_vector <= std_logic_vector(to_unsigned(5*h_count + 2*v_count + 10*frame_count, 16));
169: --      blue_vector <= std_logic_vector(to_unsigned(3*v_count - 2*h_count + 6*frame_count, 16));
170:
171:
172:         frame_vector <= std_logic_vector(to_unsigned(frame_count, 16));
173:
174: --      red <= to_integer(unsigned(red_vector(12 downto 9)));
175: --      green <= to_integer(unsigned(green_vector(12 downto 9)));
176: --      blue <= to_integer(unsigned(blue_vector(12 downto 9)));
177:
178:
179: --      red <= to_integer(unsigned(red_vector(12 downto 9)));
180: --      green <= to_integer(unsigned(green_vector(12 downto 9)));
181: --      blue <= to_integer(unsigned(blue_vector(12 downto 9)));
182:
183:         o_mport_miso.wr.clk <= i_clk;
184:         o_mport_miso.wr.en <= '0';
185:
186:
187:         o_mport_miso.cmd.clk <= i_clk;
188:
189:         process(i_clk)
190:         begin
191:                 if ( rising_edge(i_clk) ) then
192:                         cmd_empty_d1 <= i_mport_mosi.cmd.empty;
193:                 end if;
194:         end process;
195:
196:         process(i_clk)
197:         begin
198:                 if ( rising_edge(i_clk) ) then
199:                         if ( frame_counter_enable = '1' ) then
200:                                 addr_latch <= (others => '0');
```

```
  201:                        elsif ( h_sync = '1' ) then
  202:                                h_fetch_count <= 0;
  203:                        elsif ( i_mport_mosi.cmd.empty = '1' and cmd_empty_d1 = '1' and h_fetch_count < H_ACTIVE_WIDTH ) then -- and h_active = '1' )
 then
  204:                        --elsif ( i_mport_mosi.cmd.empty = '1' and cmd_empty_d1 = '1' and  h_fetch_count < H_ACTIVE_WIDTH and to_integer(unsigned(i_m
port_mosi.rd.count)) < (2*LINE_BURST_LENGTH) and h_active = '1' ) then
  205:                                o_mport_miso.cmd.en <= '1';
  206:                                o_mport_miso.cmd.instr <= "011";
  207:                                --o_mport_miso.cmd.bl <= "001111";
  208:                                o_mport_miso.cmd.bl <= std_logic_vector(to_unsigned((LINE_BURST_LENGTH-1), 6));
  209:                                o_mport_miso.cmd.byte_addr <= "0000" & addr_latch(25 downto 0);
  210:                                addr_latch <= std_logic_vector(to_unsigned(LINE_BURST_LENGTH*4+to_integer(unsigned(addr_latch)),26));
  211:                                h_fetch_count <= LINE_BURST_LENGTH + h_fetch_count;
  212:                        else
  213:                                o_mport_miso.cmd.en <= '0';
  214:                        end if;
  215:                end if;
  216:        end process;
  217:
  218:
  219:
  220:
  221:        o_mport_miso.rd.clk <= i_clk;
  222:
  223:        process(i_clk)
  224:        begin
  225:                if ( rising_edge(i_clk) ) then
  226:                        --if ( h_subframe_active = '1' and v_subframe_active = '1' ) then
  227:                        if ( h_active = '1' and v_active = '1' ) then
  228:
  229:                                o_mport_miso.rd.en <= '1';
  230:
  231:                                o_vga_mosi.red <= i_mport_mosi.rd.data(15 downto 12);
  232:                                o_vga_mosi.green <= i_mport_mosi.rd.data(10 downto 7);
  233:                                o_vga_mosi.blue <= i_mport_mosi.rd.data(4 downto 1);
  234:
  235:
  236: --                             o_vga_mosi.red <= std_logic_vector(to_unsigned(red,4));
  237: --                             o_vga_mosi.green <= std_logic_vector(to_unsigned(green,4));
  238: --                             o_vga_mosi.blue <= std_logic_vector(to_unsigned(blue,4));
  239:
  240:                        else
  241:
  242:                                o_mport_miso.rd.en <= '0';
  243:
  244:                                o_vga_mosi.red <= (others => '0');
  245:                                o_vga_mosi.green <= (others => '0');
  246:                                o_vga_mosi.blue <= (others => '0');
  247:                        end if;
  248:                end if;
  249:        end process;
  250:
  251:
  252:        process(i_clk)
  253:        begin
  254:                if ( rising_edge(i_clk) ) then
  255:                        o_vga_mosi.hsync <= h_sync;
  256:                        o_vga_mosi.vsync <= v_sync;
  257:                end if;
  258:        end process;
  259:
  260:
  261:
  262:        h_counter : cpt_upcounter
  263:        generic map (
  264:                INIT => 0
  265:        )
  266:        port map (
  267:                i_clk => i_clk,
  268:                i_enable => '1',
  269:                i_lowest => H_FRAME_FIRST,
  270:                i_highest => H_FRAME_LAST,
  271:                i_increment => 1,
  272:                i_clear => reset,
  273:                i_preset => '0',
  274:                o_count => h_count,
  275:                o_carry => open
  276:        );
  277:
  278:        process(i_clk)
  279:        begin
  280:                if ( rising_edge(i_clk) ) then
  281:                        if ( reset = '0' and h_count >= H_ACTIVE_FIRST and h_count <= H_ACTIVE_LAST ) then
  282:                                h_active <= '1';
  283:                        else
  284:                                h_active <= '0';
  285:                        end if;
  286:                end if;
  287:        end process;
  288:
  289:        process(i_clk)
  290:        begin
  291:                if ( rising_edge(i_clk) ) then
  292:                        if ( reset = '0' and h_count >= H_SUBFRAME_FIRST and h_count <= H_SUBFRAME_LAST ) then
  293:                                h_subframe_active <= '1';
  294:                        else
  295:                                h_subframe_active <= '0';
  296:                        end if;
  297:                end if;
  298:        end process;
```

```
299:
300:        process(i_clk)
301:        begin
302:                if ( rising_edge(i_clk) ) then
303:                        if ( reset = '0' and h_count >= H_SYNC_FIRST and h_count <= H_SYNC_LAST ) then
304:                                h_sync <= '1';
305:                        else
306:                                h_sync <= '0';
307:                        end if;
308:                end if;
309:        end process;
310:
311:
312:
313:
314:        process(i_clk)
315:        begin
316:                if ( rising_edge(i_clk) ) then
317:                        if ( h_count = H_FRAME_LAST ) then
318:                                v_counter_enable <= '1';
319:                        else
320:                                v_counter_enable <= '0';
321:                        end if;
322:                end if;
323:        end process;
324:
325:        v_counter : cpt_upcounter
326:        generic map (
327:                INIT => 0
328:        )
329:        port map (
330:                i_clk => i_clk,
331:                i_enable => v_counter_enable,
332:                i_lowest => V_FRAME_FIRST,
333:                i_highest => V_FRAME_LAST,
334:                i_increment => 1,
335:                i_clear => reset,
336:                i_preset => '0',
337:                o_count => v_count,
338:                o_carry => open
339:        );
340:
341:        process(i_clk)
342:        begin
343:                if ( rising_edge(i_clk) ) then
344:                        if ( reset = '0' and v_count >= V_ACTIVE_FIRST and v_count <= V_ACTIVE_LAST ) then
345:                                v_active <= '1';
346:                        else
347:                                v_active <= '0';
348:                        end if;
349:                end if;
350:        end process;
351:
352:        process(i_clk)
353:        begin
354:                if ( rising_edge(i_clk) ) then
355:                        if ( reset = '0' and v_count >= V_SUBFRAME_FIRST and v_count <= V_SUBFRAME_LAST ) then
356:                                v_subframe_active <= '1';
357:                        else
358:                                v_subframe_active <= '0';
359:                        end if;
360:                end if;
361:        end process;
362:
363:        process(i_clk)
364:        begin
365:                if ( rising_edge(i_clk) ) then
366:                        if ( reset = '0' and v_count >= V_SYNC_FIRST and v_count <= V_SYNC_LAST ) then
367:                                v_sync <= '1';
368:                        else
369:                                v_sync <= '0';
370:                        end if;
371:                end if;
372:        end process;
373:
374:        process(i_clk)
375:        begin
376:                if ( rising_edge(i_clk) ) then
377:                        if ( h_count = H_FRAME_LAST and v_count = V_FRAME_LAST ) then
378:                                frame_counter_enable <= '1';
379:                        else
380:                                frame_counter_enable <= '0';
381:                        end if;
382:                end if;
383:        end process;
384:
385:        frame_counter : cpt_upcounter
386:        generic map (
387:                INIT => 0
388:        )
389:        port map (
390:                i_clk => i_clk,
391:                i_enable => frame_counter_enable,
392:                i_lowest => 0,
393:                i_highest => 2**30-1,
394:                i_increment => 1,
395:                i_clear => reset,
396:                i_preset => '0',
397:                o_count => frame_count,
398:                o_carry => open
```

```
399:         );
400:
401:         process(i_clk)
402:         begin
403:                 if ( rising_edge(i_clk) ) then
404:                         if ( v_count >= 0 ) then
405:                                 frame_active <= '1';
406:                         else
407:                                 frame_active <= '0';
408:                         end if;
409:                 end if;
410:         end process;
411:
412:
413: end behavioral;
414:
```

```
    1:
    2: library ieee;
    3: use ieee.std_logic_1164.all;
    4: use ieee.numeric_std.all;
    5:
    6:
    7: library vga;
    8: use vga.pkg_vga.all;
    9:
   10: library util;
   11: use util.pkg_util.all;
   12:
   13:
   14: entity cpt_vga_test is
   15:
   16:                 generic (
   17:                         SMALL_FRAME : string := "FALSE"
   18:                 );
   19:
   20:                 port  (
   21:                         i_clk : in std_logic;
   22:                         i_enable : in std_logic;
   23:                         i_vga_test_mode : in std_logic;
   24:                         o_vga_mosi : out typ_vga_mosi
   25:                 );
   26:
   27: end cpt_vga_test;
   28:
   29:
   30: architecture behavioral of cpt_vga_test is
   31:
   32:         function f(v1:integer; v2:integer) return integer is
   33:                 begin
   34:                 if (SMALL_FRAME = "FALSE") then
   35:                   return v1;
   36:                 else
   37:                   return v2;
   38:                 end if;
   39:    end function;
   40:
   41:
   42:         -- ----------------------------------
   43:         -- Characteristic timing constants
   44:         -- ----------------------------------
   45:
   46:         constant H_ACTIVE_WIDTH : integer := f(1280, 12);
   47:         constant H_FRONTPORCH_WIDTH : integer := f(48, 2);
   48:         constant H_SYNC_WIDTH : integer := f(112, 1);
   49:         constant H_BACKPORCH_WIDTH : integer := f(248, 2);
   50:
   51:         constant V_ACTIVE_WIDTH : integer := f(1024, 12);
   52:         constant V_FRONTPORCH_WIDTH : integer := f(1, 1);
   53:         constant V_SYNC_WIDTH : integer := f(3, 1);
   54:         constant V_BACKPORCH_WIDTH : integer := f(38, 2);
   55:
   56:
   57:         -- ----------------------------------------------------
   58:         -- Derived timing constants (do not modify without reason)
   59:         -- ----------------------------------------------------
   60:
   61:         constant H_ACTIVE_FIRST : integer := 0;
   62:         constant H_ACTIVE_LAST : integer := H_ACTIVE_FIRST + H_ACTIVE_WIDTH - 1;
   63:
   64:         constant H_FRONTPORCH_FIRST : integer := H_ACTIVE_LAST + 1;
   65:         constant H_FRONTPORCH_LAST : integer := H_FRONTPORCH_FIRST + H_FRONTPORCH_WIDTH - 1;
   66:
   67:         constant H_SYNC_FIRST : integer := H_FRONTPORCH_LAST + 1;
   68:         constant H_SYNC_LAST : integer := H_SYNC_FIRST + H_SYNC_WIDTH - 1;
   69:
   70:         constant H_BACKPORCH_FIRST : integer := H_SYNC_LAST + 1;
   71:         constant H_BACKPORCH_LAST : integer := H_BACKPORCH_FIRST + H_BACKPORCH_WIDTH - 1;
   72:
   73:         constant H_BLANK_FIRST : integer := H_FRONTPORCH_FIRST;
   74:         constant H_BLANK_LAST : integer := H_BACKPORCH_LAST;
   75:
   76:         constant H_FRAME_FIRST : integer := H_ACTIVE_FIRST;
   77:         constant H_FRAME_LAST : integer := H_BACKPORCH_LAST;
   78:
   79:         constant V_ACTIVE_FIRST : integer := 0;
   80:         constant V_ACTIVE_LAST : integer := V_ACTIVE_FIRST + V_ACTIVE_WIDTH - 1;
   81:
   82:         constant V_FRONTPORCH_FIRST : integer := V_ACTIVE_LAST + 1;
   83:         constant V_FRONTPORCH_LAST : integer := V_FRONTPORCH_FIRST + V_FRONTPORCH_WIDTH - 1;
   84:
   85:         constant V_SYNC_FIRST : integer := V_FRONTPORCH_LAST + 1;
   86:         constant V_SYNC_LAST : integer := V_SYNC_FIRST + V_SYNC_WIDTH - 1;
   87:
   88:         constant V_BACKPORCH_FIRST : integer := V_SYNC_LAST + 1;
   89:         constant V_BACKPORCH_LAST : integer := V_BACKPORCH_FIRST + V_BACKPORCH_WIDTH - 1;
   90:
   91:         constant V_BLANK_FIRST : integer := V_FRONTPORCH_FIRST;
   92:         constant V_BLANK_LAST : integer := V_BACKPORCH_LAST;
   93:
   94:         constant V_FRAME_FIRST : integer := V_ACTIVE_FIRST;
   95:         constant V_FRAME_LAST : integer := V_BACKPORCH_LAST;
   96:
   97:
   98:         -- Video timing generation
   99:         signal h_count : integer := 0;
  100:         signal h_active : std_logic := '1';
```

```
101:         signal h_sync : std_logic := '0';
102:
103:         signal v_counter_enable : std_logic := '1';
104:         signal v_count : integer := 0;
105:         signal v_active : std_logic := '1';
106:         signal v_sync : std_logic := '0';
107:
108:         signal frame_counter_enable : std_logic := '1';
109:         signal frame_count : integer := 0;
110:         signal frame_active : std_logic := '1';
111:
112:         signal reset : std_logic := '0';
113:
114:
115:         signal red : integer := 0;
116:         signal green : integer := 0;
117:         signal blue : integer := 0;
118:
119:
120:         signal red_vector : std_logic_vector(15 downto 0) := (others => '0');
121:         signal green_vector : std_logic_vector(15 downto 0) := (others => '0');
122:         signal blue_vector : std_logic_vector(15 downto 0) := (others => '0');
123:
124:
125:         signal frame_vector : std_logic_vector(15 downto 0) := (others => '0');
126:
127: begin
128:
129:         reset <= not i_enable;
130:
131:
132:         red_vector <= std_logic_vector(to_unsigned(4*frame_count - 3*h_count - 5*v_count, 16));
133:         green_vector <= std_logic_vector(to_unsigned(5*h_count + 2*v_count + 10*frame_count, 16));
134:         blue_vector <= std_logic_vector(to_unsigned(3*v_count - 2*h_count + 6*frame_count, 16));
135:
136:
137:         frame_vector <= std_logic_vector(to_unsigned(frame_count, 16));
138:
139:         red <= to_integer(unsigned(red_vector(12 downto 9)));
140:         green <= to_integer(unsigned(green_vector(12 downto 9)));
141:         blue <= to_integer(unsigned(blue_vector(12 downto 9)));
142:
143:         process(i_clk)
144:         begin
145:                 if ( rising_edge(i_clk) ) then
146:                         if ( h_active = '1' and v_active = '1' ) then
147:                                 if ( i_vga_test_mode = '0' ) then
148:                                         if ( h_count mod 2 = 0 ) then
149:                                                 o_vga_mosi.red <= (others => '0');
150:                                                 o_vga_mosi.green <= (others => '0');
151:                                                 o_vga_mosi.blue <= (others => '0');
152:                                         else
153:                                                 o_vga_mosi.red <= (others => '1');
154:                                                 o_vga_mosi.green <= (others => '1');
155:                                                 o_vga_mosi.blue <= (others => '1');
156:                                         end if;
157:                                 else
158:                                         o_vga_mosi.red <= std_logic_vector(to_unsigned(red,4));
159:                                         o_vga_mosi.green <= std_logic_vector(to_unsigned(green,4));
160:                                         o_vga_mosi.blue <= std_logic_vector(to_unsigned(blue,4));
161:                                 end if;
162:                         else
163:                                 o_vga_mosi.red <= (others => '0');
164:                                 o_vga_mosi.green <= (others => '0');
165:                                 o_vga_mosi.blue <= (others => '0');
166:                         end if;
167:                 end if;
168:         end process;
169:
170:
171:         process(i_clk)
172:         begin
173:                 if ( rising_edge(i_clk) ) then
174:                         o_vga_mosi.hsync <= h_sync;
175:                         o_vga_mosi.vsync <= v_sync;
176:                 end if;
177:         end process;
178:
179:
180:
181:         h_counter : cpt_upcounter
182:         generic map (
183:                 INIT => 0
184:         )
185:         port map (
186:                 i_clk => i_clk,
187:                 i_enable => '1',
188:                 i_lowest => H_FRAME_FIRST,
189:                 i_highest => H_FRAME_LAST,
190:                 i_increment => 1,
191:                 i_clear => reset,
192:                 i_preset => '0',
193:                 o_count => h_count,
194:                 o_carry => open
195:         );
196:
197:         process(i_clk)
198:         begin
199:                 if ( rising_edge(i_clk) ) then
200:                         if ( reset = '0' and h_count >= H_ACTIVE_FIRST and h_count <= H_ACTIVE_LAST ) then
```

```
201:                                    h_active <= '1';
202:                            else
203:                                    h_active <= '0';
204:                            end if;
205:                    end if;
206:            end process;
207:
208:            process(i_clk)
209:            begin
210:                    if ( rising_edge(i_clk) ) then
211:                            if ( reset = '0' and h_count >= H_SYNC_FIRST and h_count <= H_SYNC_LAST ) then
212:                                    h_sync <= '1';
213:                            else
214:                                    h_sync <= '0';
215:                            end if;
216:                    end if;
217:            end process;
218:
219:
220:
221:
222:            process(i_clk)
223:            begin
224:                    if ( rising_edge(i_clk) ) then
225:                            if ( h_count = H_FRAME_LAST ) then
226:                                    v_counter_enable <= '1';
227:                            else
228:                                    v_counter_enable <= '0';
229:                            end if;
230:                    end if;
231:            end process;
232:
233:            v_counter : cpt_upcounter
234:            generic map (
235:                    INIT => 0
236:            )
237:            port map (
238:                    i_clk => i_clk,
239:                    i_enable => v_counter_enable,
240:                    i_lowest => V_FRAME_FIRST,
241:                    i_highest => V_FRAME_LAST,
242:                    i_increment => 1,
243:                    i_clear => reset,
244:                    i_preset => '0',
245:                    o_count => v_count,
246:                    o_carry => open
247:            );
248:
249:            process(i_clk)
250:            begin
251:                    if ( rising_edge(i_clk) ) then
252:                            if ( reset = '0' and v_count >= V_ACTIVE_FIRST and v_count <= V_ACTIVE_LAST ) then
253:                                    v_active <= '1';
254:                            else
255:                                    v_active <= '0';
256:                            end if;
257:                    end if;
258:            end process;
259:
260:            process(i_clk)
261:            begin
262:                    if ( rising_edge(i_clk) ) then
263:                            if ( reset = '0' and v_count >= V_SYNC_FIRST and v_count <= V_SYNC_LAST ) then
264:                                    v_sync <= '1';
265:                            else
266:                                    v_sync <= '0';
267:                            end if;
268:                    end if;
269:            end process;
270:
271:            process(i_clk)
272:            begin
273:                    if ( rising_edge(i_clk) ) then
274:                            if ( h_count = H_FRAME_LAST and v_count = V_FRAME_LAST ) then
275:                                    frame_counter_enable <= '1';
276:                            else
277:                                    frame_counter_enable <= '0';
278:                            end if;
279:                    end if;
280:            end process;
281:
282:            frame_counter : cpt_upcounter
283:            generic map (
284:                    INIT => 0
285:            )
286:            port map (
287:                    i_clk => i_clk,
288:                    i_enable => frame_counter_enable,
289:                    i_lowest => 0,
290:                    i_highest => 2**30-1,
291:                    i_increment => 1,
292:                    i_clear => reset,
293:                    i_preset => '0',
294:                    o_count => frame_count,
295:                    o_carry => open
296:            );
297:
298:            process(i_clk)
299:            begin
300:                    if ( rising_edge(i_clk) ) then
```

```
301:                        if ( v_count >= 0 ) then
302:                                frame_active <= '1';
303:                        else
304:                                frame_active <= '0';
305:                        end if;
306:                end if;
307:        end process;
308:
309:
310: end behavioral;
311:
```

```
--
-- VGA package
--

library ieee;
use ieee.std_logic_1164.all;

library mctl;
use mctl.pkg_mctl.all;

package pkg_vga is

	-- SXGA 1280x1024 @ 60Hz
	-- http://tinyvga.com/vga-timing/1280x1024@60Hz
	constant PIXELS_PER_LINE : integer := 1280;	-- Total number of pixels per line
	constant MAX_LINES : integer := 1024;		-- Total number of lines on screen
	constant H_FP : integer := 48;		-- Horizontal front porch
	constant H_PULSE : integer := 112;	-- Horizontal sync pulse
	constant H_BP : integer := 248;		-- Horizontal back porch
	constant V_FP : integer := 1;		-- Vertical front porch
	constant V_PULSE : integer := 3;	-- Vertical sync pulse
	constant V_BP : integer := 38;		-- Vertical back porch


	type typ_vga_mosi is record
		red : std_logic_vector(3 downto 0);
		green : std_logic_vector(3 downto 0);
		blue : std_logic_vector(3 downto 0);
		hsync : std_logic;
		vsync : std_logic;
	end record;

	constant init_vga_mosi : typ_vga_mosi := (
		red => (others => '0'),
		green => (others => '0'),
		blue => (others => '0'),
		hsync => '0',
		vsync => '0'
	);


	component cpt_vga is
		port (
			i_clk : in std_logic;
			i_enable : in std_logic;
			i_linebuf_data : in std_logic_vector(15 downto 0);	-- Input RGB data
			o_line_start : out std_logic;	-- Beginning of line indicator flag
			o_pixel_number : out integer range -2048 to 2047;
			o_line_number : out integer range -1024 to 1023;
			o_frame_number : out integer range 0 to 3;
			o_vga_mosi : out typ_vga_mosi := init_vga_mosi  -- Output RGB data, hsync, vsync
		);
	end component;

	component cpt_linebuf is
		 port (
			i_enable : in std_logic;
			i_clk : in std_logic;

			i_frame_addr0 : in std_logic_vector (28 downto 0);
			i_frame_addr1 : in std_logic_vector (28 downto 0);
			i_frame_addr2 : in std_logic_vector (28 downto 0);
			i_frame_addr3 : in std_logic_vector (28 downto 0);
			i_frame_number : in integer := 0;

			i_line_start : in std_logic;
			i_mid_line_offset : in integer range -(2**24) to (2**24)-1 := 0;
			i_line_number : in integer range -1024 to 1023 := 0;

			i_burst_length : in std_logic_vector (5 downto 0);
			i_pixel_number : in integer range -2048 to 2047;

			i_mport_mosi : in typ_mctl_mport_mosi;
			o_mport_miso : out typ_mctl_mport_miso;

			o_linebuf_data : out std_logic_vector (15 downto 0)
		);
	end component;


	component cpt_vga_test is
		port (
			i_clk : in std_logic;
			i_enable : in std_logic;
			i_vga_test_mode : in std_logic;
			o_vga_mosi : out typ_vga_mosi
		);
	end component;

	component cpt_vga_fixed is
		generic (
			SMALL_FRAME : string := "FALSE"
		);
		port (
			i_clk : in std_logic;
			i_enable : in std_logic;
			i_mport_mosi : in typ_mctl_mport_mosi;
			o_mport_miso : out typ_mctl_mport_miso;
			o_vga_mosi : out typ_vga_mosi
		);
```

```
101:         end component;
102:
103: end pkg_vga;
104:
105: package body pkg_vga is
106:
107: end pkg_vga;
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5:
  6: use ieee.numeric_std.all;
  7:
  8: library vga;
  9: use vga.pkg_vga.all;
 10:
 11: entity test_linebuf is
 12: end test_linebuf;
 13:
 14: architecture Behavioral of test_linebuf is
 15:
 16:         signal clk : std_logic;
 17:         signal enable : std_logic;
 18:         signal linebuf_data : std_logic_vector(15 downto 0);
 19:         signal line_start : std_logic;
 20:         signal pixel_number : integer range -2048 to 2047;
 21:         signal line_number : integer range -1024 to 1023;
 22:         signal frame_number : integer range 0 to 3;
 23:         signal vga_mosi : typ_vga_mosi;
 24:
 25:
 26:
 27:         component cpt_linebuf is
 28:                  port (
 29:                         signal enable : std_logic;
 30:                         signal clk : std_logic;
 31:
 32:                         signal frame_addr0 : std_logic_vector (25 downto 0);
 33:                         signal frame_addr1 : std_logic_vector (25 downto 0);
 34:                         signal frame_addr2 : std_logic_vector (25 downto 0);
 35:                         signal frame_addr3 : std_logic_vector (25 downto 0);
 36:                         signal frame_number : integer := 0;
 37:
 38:                         signal line_start : std_logic;
 39:                         signal mid_line_offset : integer range -(2**24) to ((2**24)-1) := 0;
 40:                         signal line_number : integer range -1024 to 1023 := 0;
 41:
 42:                         signal burst_length : std_logic_vector (5 downto 0);
 43:                         signal pixel_number : std_logic_vector (11 downto 0);
 44:
 45:                         signal mport_mosi : typ_mctl_mport_mosi;
 46:                         signal mport_miso : typ_mctl_mport_miso;
 47:
 48:                         signal linebuf_data : std_logic_vector (15 downto 0)
 49:                 );
 50:         end component;
 51:
 52:
 53:
 54:
 55:
 56: i_enable => enable,
 57:                 i_clk => clk,
 58:
 59:                 i_frame_addr0 => frame_addr0,
 60:                 i_frame_addr1 => frame_addr1,
 61:                 i_frame_addr2 => frame_addr2,
 62:                 i_frame_addr3 => frame_addr3,
 63:                 i_frame_number => frame_number,
 64:
 65:                 i_line_start => line_start,
 66:                 i_mid_line_offset => mid_line_offset,
 67:                 i_line_number => line_number,
 68:
 69:                 i_burst_length => burst_length,
 70:                 i_pixel_number => pixel_number,
 71:
 72:                 i_mport_mosi => mport_mosi,
 73:                 o_mport_miso mport_miso,
 74:
 75:                 o_linebuf_data => linebuf_data,
 76:
 77:
 78:
 79:
 80: begin
 81:
 82:         process
 83:         begin
 84:                 enable <= '1';
 85:                 wait for 1 us;
 86:                 enable <= '0';
 87:                 wait for 1 us;
 88:                 enable <= '1';
 89:                 wait;
 90:         end process;
 91:
 92:         process
 93:         begin
 94:                 clk <= '0';
 95:                 wait for 4.63 ns;
 96:                 clk <= '1';
 97:                 wait for 4.63 ns;
 98:         end process;
 99:
100:
```

```
101:
102:         linebuf : cpt_linebuf
103:          port map(
104:                 i_enable => enable,
105:                 i_clk => clk,
106:
107:                 i_frame_addr0 => frame_addr0,
108:                 i_frame_addr1 => frame_addr1,
109:                 i_frame_addr2 => frame_addr2,
110:                 i_frame_addr3 => frame_addr3,
111:                 i_frame_number => frame_number,
112:
113:                 i_line_start => line_start,
114:                 i_mid_line_offset => mid_line_offset,
115:                 i_line_number => line_number,
116:
117:                 i_burst_length => burst_length,
118:                 i_pixel_number => pixel_number,
119:
120:                 i_mport_mosi => mport_mosi,
121:                 o_mport_miso mport_miso,
122:
123:                 o_linebuf_data => linebuf_data,
124:         );
125:
126: end Behavioral;
127:
```

```
  1:
  2: library ieee;
  3: use ieee.std_logic_1164.all;
  4:
  5:
  6: use ieee.numeric_std.all;
  7:
  8: library mctl;
  9: use mctl.pkg_mctl.all;
 10:
 11: library vga;
 12: use vga.pkg_vga.all;
 13:
 14: entity test_linebuf_to_vga is
 15: end test_linebuf_to_vga;
 16:
 17:
 18: architecture Behavioral of test_linebuf_to_vga is
 19:
 20:         signal enable : std_logic;
 21:         signal clk : std_logic;
 22:
 23:
 24:
 25:         -- mport-linebuf common signals
 26:         signal mport_mosi : typ_mctl_mport_mosi;
 27:         signal mport_miso : typ_mctl_mport_miso;
 28:
 29:
 30:
 31:         -- linebuf-VGA common signals
 32:         signal line_start : std_logic := '0';
 33:         signal linebuf_data : std_logic_vector(15 downto 0);
 34:         signal pixel_number : integer range -2048 to 2047 := 0;
 35:         signal line_number : integer range -1024 to 1023 := 0;
 36:         signal frame_number : integer range 0 to 3 := 0;
 37:
 38:
 39:         -- VGA signals
 40:         signal vga_mosi : typ_vga_mosi;
 41:
 42:
 43:         -- Linebuf signals
 44:         signal frame_addr0 : std_logic_vector (25 downto 0) := x"000000" & "00";
 45:         signal frame_addr1 : std_logic_vector (25 downto 0) := x"010000" & "00";
 46:         signal frame_addr2 : std_logic_vector (25 downto 0) := x"020000" & "00";
 47:         signal frame_addr3 : std_logic_vector (25 downto 0) := x"030000" & "00";
 48:         signal mid_line_offset : integer range -(2**24) to ((2**24)-1) := 0;
 49:         signal burst_length : std_logic_vector (5 downto 0) := "001111";
 50:
 51:
 52:    constant clk_period : time := 9.259 ns;
 53:
 54:
 55: begin
 56:
 57:         process
 58:         begin
 59:                 enable <= '1';
 60:                 wait for 1 us;
 61:                 enable <= '0';
 62:                 wait for 1 us;
 63:                 enable <= '1';
 64:                 wait;
 65:         end process;
 66:
 67:         process
 68:         begin
 69:                 clk <= '0';
 70:                 wait for clk_period/2;
 71:                 clk <= '1';
 72:                 wait for clk_period/2;
 73:         end process;
 74:
 75:
 76:         mport_mosi.cmd.empty <= '1';
 77:         mport_mosi.cmd.full <= '0';
 78:
 79:         mport_mosi.rd.empty <= '1';
 80:         mport_mosi.rd.full <= '0';
 81:         mport_mosi.rd.count <= (others => '0');
 82:         mport_mosi.rd.overflow <= '0';
 83:         mport_mosi.rd.error <= '0';
 84:         mport_mosi.rd.data <= (others => '0');
 85:
 86:         mport_mosi.wr.empty <= '1';
 87:         mport_mosi.wr.full <= '0';
 88:         mport_mosi.wr.count <=  (others => '0');
 89:         mport_mosi.wr.underrun <= '0';
 90:         mport_mosi.wr.error <= '0';
 91:
 92:
 93:         linebuf : cpt_linebuf
 94:          port map(
 95:                 i_enable => enable,
 96:                 i_clk => clk,
 97:
 98:                 i_frame_addr0 => frame_addr0,
 99:                 i_frame_addr1 => frame_addr1,
100:                 i_frame_addr2 => frame_addr2,
```

```
101:                 i_frame_addr3 => frame_addr3,
102:                 i_frame_number => frame_number,
103:
104:                 i_line_start => line_start,
105:                 i_mid_line_offset => mid_line_offset,
106:                 i_line_number => line_number,
107:
108:                 i_burst_length => burst_length,
109:                 i_pixel_number => pixel_number,
110:
111:                 i_mport_mosi => mport_mosi,
112:                 o_mport_miso => mport_miso,
113:
114:                 o_linebuf_data => linebuf_data
115:         );
116:
117:         vga : cpt_vga
118:         port map (
119:                 i_clk => clk,
120:                 i_enable => enable,
121:                 i_linebuf_data => linebuf_data, -- Input RGB data
122:                 o_line_start => line_start,     -- Beginning of line indicator flag
123:                 o_pixel_number => pixel_number,
124:                 o_line_number => line_number,
125:                 o_frame_number => frame_number,
126:                 o_vga_mosi =>   vga_mosi -- Output RGB data, hsync, vsync
127:         );
128:
129: end Behavioral;
130:
```

```
 1:
 2: library ieee;
 3: use ieee.std_logic_1164.all;
 4:
 5:
 6: use ieee.numeric_std.all;
 7:
 8: library vga;
 9: use vga.pkg_vga.all;
10:
11: entity test_vga is
12: end test_vga;
13:
14: architecture Behavioral of test_vga is
15:
16:         signal clk : std_logic;
17:         signal enable : std_logic;
18:         signal linebuf_data : std_logic_vector(15 downto 0);
19:         signal line_start : std_logic;
20:         signal pixel_number : integer range -2048 to 2047;
21:         signal line_number : integer range -1024 to 1023;
22:         signal frame_number : integer range 0 to 3;
23:         signal vga_mosi : typ_vga_mosi;
24:
25: begin
26:
27:         process
28:         begin
29:                 enable <= '1';
30:                 wait for 1 us;
31:                 enable <= '0';
32:                 wait for 1 us;
33:                 enable <= '1';
34:                 wait;
35:         end process;
36:
37:         process
38:         begin
39:                 clk <= '0';
40:                 wait for 4.63 ns;
41:                 clk <= '1';
42:                 wait for 4.63 ns;
43:         end process;
44:
45:
46:         process(clk)
47:         begin
48:                 if rising_edge(clk) then
49:                         linebuf_data(15 downto 12) <= std_logic_vector(to_unsigned((pixel_number/256) mod 16, 4));
50:                         linebuf_data(10 downto 7) <= std_logic_vector(to_unsigned((pixel_number/16) mod 16, 4));
51:                         linebuf_data(4 downto 1) <= std_logic_vector(to_unsigned(pixel_number mod 16, 4));
52:                 end if;
53:         end process;
54:
55:
56:         linebuf_data(11) <= '0';
57:         linebuf_data(6) <= '0';
58:         linebuf_data(5) <= '0';
59:         linebuf_data(0) <= '0';
60:
61:         vga : cpt_vga
62:         port map (
63:                 i_clk => clk,
64:                 i_enable => enable,
65:                 i_linebuf_data => linebuf_data, -- Input RGB data
66:                 o_line_start => line_start,     -- Beginning of line indicator flag
67:                 o_pixel_number => pixel_number,
68:                 o_line_number => line_number,
69:                 o_frame_number => frame_number,
70:                 o_vga_mosi =>   vga_mosi -- Output RGB data, hsync, vsync
71:         );
72:
73:
74: end Behavioral;
75:
```