```
-------------------------------------------------------------------------------
-- $Archive:: /VHDL/product/c6416tdsk/c6416tdsk.vhd                           $
-- $Revision:: 7                                                              $
-- $Date:: 8/15/04 7:37a                                                      $
-- $Author:: Tonyc                                                            $
--
--
-- Copyright (c) 2002,2003,2004, Spectrum Digital Incorporated
-- All rights reserved
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-- Start the real code
-------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity c6416tdsk is
  port
  (
    CLKIN        : in    std_logic; -- 12 MHz clock in
    OPT_CLK1     : in    std_logic; -- 25 MHz clock in
    OPT_CLK2     : in    std_logic; -- 8 MHz clock in
    EMU_RSTn     : in    std_logic; -- Emulator reset from USB block
    PONRSn       : in    std_logic; -- Power on reset from voltage supervisor
    PUSHBRS      : in    std_logic; -- Push button reset
    HPIRSn       : in    std_logic; -- HPI reset from optional DC

    -- DSP Memory interface signals for internal CPLD registers.  From C64xx B port.
    DSP_DQ       : inout std_logic_vector( 7 downto 0 );    -- DSP Data bus
    DSP_ADDR     : in    std_logic_vector( 2 downto 0 );    -- DSP Address bus
    DSP_CSn      : in    std_logic;                         -- DSP Chip select
    DSP_WEn      : in    std_logic;                         -- DSP Write strobe
    DSP_REn      : in    std_logic;                         -- DSP Read strobe
    DSP_OEn      : in    std_logic;                         -- DSP Output enable
    DSP_RSn      : out   std_logic;                         -- DSP Reset

    -- DSP Memory interface signals that control the daughter card. From C64x A port.
    DSP_DC_CS0n  : in    std_logic;                 -- DSP DC Chip select (ACE2#)
    DSP_DC_CS1n  : in    std_logic;                 -- DSP DC Chip select (ACE3#)
    DSP_DC_WEn   : in    std_logic;                 -- DSP DC Write strobe
    DSP_DC_REn   : in    std_logic;                 -- DSP DC Read strobe
    DSP_DC_OEn   : in    std_logic;                 -- DSP DC Output enable
    DSP_CLKMODE0 : out    std_logic;                -- DSP CLKMODE0
    DSP_CLKMODE1 : out    std_logic;                -- DSP CLKMODE1
```

```
    -- User/Board Support
    USER_SW      : in    std_logic_vector( 3 downto 0 );  -- User swtiches
    USER_LED     : out   std_logic_vector( 3 downto 0 );  -- Uwer led
    PWB_REV      : in    std_logic_vector( 2 downto 0 );  -- PWB revision

    -- Daughter Card Support
    DC_STAT      : in    std_logic_vector( 1 downto 0 );   -- DC Status
    DC_CNTL      : out   std_logic_vector( 1 downto 0 );   -- DC Control
    DC_DBUF_DIR  : out   std_logic;                  -- DC Data buffer direction
    DC_DBUF_OEn  : out   std_logic;                  -- DC Data buffer output enable
    DC_CNTL_OEn  : out   std_logic;                  -- DC Control buffer enable
    DC_DETn      : in    std_logic;                  -- DC Detect
    DC_RESETn    : out   std_logic;                  -- DC Reset

    -- McBSP Multiplexer Control
    MCBSP_SELA   : out   std_logic;                  -- Codec/DC McBsp 1 mux cntl
    MCBSP_SELB   : out   std_logic;                  -- Codec/DC McBsp 1 mux cntl
    MCBSP2_EN    : in    std_logic;                  -- McBSP2 enable from PCI DC

    -- Misc. Stuff
    BRD_RSn      : out   std_logic;                  -- Board reset
    DSP_RSn_LED  : out   std_logic;                  -- DSP reset led
    FLASH_PAGE   : out   std_logic;                  -- Flash Address 19
    CPLD_CLK_OUT : out   std_logic;                  -- Place holder

    -- CLOCKING FOR EMIF AND CPU
    TEST_PIN        : out   std_logic;
    DSP_PLL_CLK     : out   std_logic;
    EMIF_PLL_CLK    : out   std_logic;
    DSPPLL_ENABLE   : in   std_logic;
    DSPPLL_SELECT   : in   std_logic;
    DSPPLL_SW       : in   std_logic_vector( 3 downto 0 );
    EMIF_PLL_S      : out  std_logic_vector( 1 downto 0 );
    CPU_DSPPLL_S    : out  std_logic_vector( 1 downto 0 ));


   end c6416tdsk;

--------------------------------------------------------------------------------
-- Include standard librariess
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
```

```
-- use work.std_arith.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

--------------------------------------------------------------------------------
-- Include fpga specifics here if required.
-- act3  is for Actel 54sx devices.  We normally use this for the hardwired
-- clock definition.
--------------------------------------------------------------------------------
-- library act3;
-- use act3.components.all;

architecture behavior_c6416tdsk of c6416tdsk is

constant CPLD_VERSION         : std_logic_vector(3 downto 0)  := "0100";

--------------------------------------------------------------------------------
-- Add local components in here
--------------------------------------------------------------------------------
--component MyComponent
--port
--(
--);
--end component;

--------------------------------------------------------------------------------
-- Add signals
--------------------------------------------------------------------------------

-- CPLD Register signals
signal    CpldReg0        : std_logic_vector( 7 downto 0 );
signal    CpldReg1        : std_logic_vector( 7 downto 0 );
signal    CpldReg2        : std_logic_vector( 7 downto 0 );
signal    CpldReg3        : std_logic_vector( 7 downto 0 );
signal    CpldReg4        : std_logic_vector( 7 downto 0 );
signal    CpldReg5        : std_logic_vector( 7 downto 0 );
signal    CpldReg6        : std_logic_vector( 7 downto 0 );
signal    CpldReg7        : std_logic_vector( 7 downto 0 );
signal    OPT_CLK1_DIV2   : std_logic_vector( 1 downto 0 );
signal    OPT_CLK1_DIV    : std_logic;
signal    MuxD            : std_logic_vector( 7 downto 0 );

signal    ChipEnables     : std_logic_vector( 7 downto 0 );
signal    CpldRegCs0      : std_logic;
```

```
signal    CpldRegCs1      : std_logic;
signal    CpldRegCs2      : std_logic;
signal    CpldRegCs3      : std_logic;
signal    CpldRegCs4      : std_logic;
signal    CpldRegCs5      : std_logic;
signal    CpldRegCs6      : std_logic;
signal    CpldRegCs7      : std_logic;

signal    SystemResetn    : std_logic;
signal    CpldClkOut      : std_logic;

signal    RsClkEn         : std_logic;
signal    RsTimer         : std_logic_vector( 11 downto 0 );
signal    RsSync          : std_logic;


-------------------------------------------------------------------------------
-- The implementation
-------------------------------------------------------------------------------

begin
-------------------------------------------------------------------------------
-- Map the other components
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
-- Now define the logic
-------------------------------------------------------------------------------



  -- Generate a reset from the three sources.
  --
  SystemResetn <= '0' when     EMU_RSTn  = '0'
                            or PONRSn    = '0'
                            or PUSHBRS   = '1'
                            else '1';


  process( OPT_CLK1  )
  begin
      if( OPT_CLK1' event and OPT_CLK1 = '1' ) then
          OPT_CLK1_DIV2 <= OPT_CLK1_DIV2 + '1';
```

```
    end if;
end process;

OPT_CLK1_DIV <= OPT_CLK1_DIV2(0);

BRD_RSn     <= '0' when SystemResetn = '0' else '1';
DSP_RSn     <= '0' when SystemResetn = '0' or HPIRSn = '0' else '1';
DSP_RSn_LED <= '0' when SystemResetn = '0' or HPIRSn = '0' else '1';

-- Generate a CPLD clockout from clock input.  This is a place holder just in
-- case we need it later.
--
process( SystemResetn, CLKIN )
begin
              if SystemResetn = '0' then
      CpldClkOut <= '0';
              elsif CLKIN'event and CLKIN = '1' then
                  CpldClkOut <= not CpldClkOut;
              end if;
      end process;

      CPLD_CLK_OUT <= CpldClkOut;

-- ###########################################################################
-- Generic register addresss decode and register chip select generation.
-- VHDL compiler will reduce any unused logic so we can be verbose.
      --
      process(  DSP_ADDR )
      begin
          case DSP_ADDR( 2 downto 0) is
              when "000"  => ChipEnables  <= "00000001";
              when "001"  => ChipEnables  <= "00000010";
              when "010"  => ChipEnables  <= "00000100";
              when "011"  => ChipEnables  <= "00001000";
              when "100"  => ChipEnables  <= "00010000";
              when "101"  => ChipEnables  <= "00100000";
              when "110"  => ChipEnables  <= "01000000";
              when "111"  => ChipEnables  <= "10000000";
              when others => ChipEnables  <= "00000000";
              end case;
      end process;

CpldRegCs0 <= '1' when ChipEnables(0) = '1' and DSP_CSn = '0' else '0';
CpldRegCs1 <= '1' when ChipEnables(1) = '1' and DSP_CSn = '0' else '0';
```

```
CpldRegCs2 <= '1' when ChipEnables(2) = '1' and DSP_CSn = '0' else '0';
CpldRegCs3 <= '1' when ChipEnables(3) = '1' and DSP_CSn = '0' else '0';
CpldRegCs4 <= '1' when ChipEnables(4) = '1' and DSP_CSn = '0' else '0';
CpldRegCs5 <= '1' when ChipEnables(5) = '1' and DSP_CSn = '0' else '0';
CpldRegCs6 <= '1' when ChipEnables(6) = '1' and DSP_CSn = '0' else '0';
CpldRegCs7 <= '1' when ChipEnables(7) = '1' and DSP_CSn = '0' else '0';

      -- ###########################################################################
-- Generate logic for each CPLD register and assign it's write, read, and
-- pin values if necessary.
--
-- All CPLD register writes occur on the rising edge DSP write strobe.
      --

      -- ===========================================================================
      -- REG 0: User Register
      -- Bit 3-0   Led 3-0
      -- Bit 7-4   Switch 3-0
process( SystemResetn, DSP_WEn, CpldRegCs0, DSP_DQ )
begin
              if SystemResetn = '0' then
      CpldReg0(3 downto 0 ) <= "0000";
              elsif DSP_WEn'event and DSP_WEn = '1' then
                if( CpldRegCs0 = '1' ) then
                  CpldReg0( 3 downto 0 ) <= DSP_DQ( 3 downto 0 );
                end if;
              end if;
      end process;

CpldReg0(7 downto 4 )  <= USER_SW( 3 downto 0 );
USER_LED( 3 downto 0 ) <= not CpldReg0(3 downto 0 );

-- ===========================================================================
      -- REG 1: DC Register
      -- Bit 1-0   DC_CNTL 1-0
      -- Bit 2     NU  read 0
      -- Bit 3     DC_RESET
      -- Bit 5-4   DC_STAT 1-0
      -- Bit 6     NU  read 0
      -- Bit 7     DC_DETECT
      --
process( SystemResetn, DSP_WEn, CpldRegCs1, DSP_DQ )
begin
              if SystemResetn = '0' then
```

```
      CpldReg1(1 downto 0 ) <= "00";
      CpldReg1(3)           <= '0';  -- not Reset by default
              elsif DSP_WEn'event and DSP_WEn = '1' then
                if( CpldRegCs1 = '1' ) then
                  CpldReg1( 1 downto 0 ) <= DSP_DQ( 1 downto 0 );
                  CpldReg1(3)            <= DSP_DQ(3);
                end if;
              end if;
      end process;


CpldReg1(2)             <= '0';
CpldReg1(5 downto 4 )   <= DC_STAT( 1 downto 0 );
CpldReg1(6)             <= '0';
CpldReg1(7)             <= not DC_DETn;

DC_CNTL( 1 downto 0 )   <= CpldReg1( 1 downto 0 );

-- HPIRSn not included in the DC_RESETn equation.  This should prevent the
-- DC from holding itself in reset if HPIRSn is active.
DC_RESETn               <= '0' when    CpldReg1(3)  = '1'
                                  or SystemResetn = '0' else '1';

-- ============================================================================
      -- REG 4: Version Register
      -- Bit 2-0   PWB Revision 2-0
      -- Bit 3     NU read 0
      -- Bit 7-4   CPLD version
      CpldReg4(7 downto 0 ) <= CPLD_VERSION(3 downto 0 ) & '0' & PWB_REV(2 downto 0 );

      -- ============================================================================
      -- REG 6: Misc. Register
      -- Bit 0     McBSP1 select
      -- Bit 1     McBsp2 select
      -- Bit 2     Flash Page/Flash Address 19
      -- Bit 3     DSPPLL_SELECT1 (READ) SW3-5
      -- Bit 4     DSPPLL_SELECT2 (READ) SW3-6
      -- Bit 5     DSPPLL_SELECT3 (READ) SW3-7
      -- Bit 6     DSPPLL_SELECT4 (READ) SW3-8
      -- Bit 7     MCBSP2_EN, read DC config for PCI/McBsp2
      --
      -- MCBSP2_EN, is included so that user can implement PCI serial rom
      -- support.
      process( SystemResetn, DSP_WEn, CpldRegCs6, DSP_DQ )
```

```
begin
        if SystemResetn = '0' then
            CpldReg6(0) <= '0';
            CpldReg6(1) <= '0';
            CpldReg6(2) <= '0';


        elsif DSP_WEn'event and DSP_WEn = '1' then
          if( CpldRegCs6 = '1' ) then
                  CpldReg6(0) <= DSP_DQ(0);
               CpldReg6(1) <= DSP_DQ(1);
               CpldReg6(2) <= DSP_DQ(2);
           end if;
        end if;
end process;

-- Low  = DC/UTOPIA
-- High = Codec
MCBSP_SELA  <= '1' when CpldReg6(0) = '0' else '0';

-- Low  = DC/PCI-SERIAL ROM
-- High = Codec
--
-- If MCBSP2_EN is low then user is requesting use of McBsp2 for
-- PCI serial ROM support.  In this case disable Codec support.
-- The user can reenable Codec support by controlling McBSP2_EN
-- on the daughter card after boot is complete.
--
MCBSP_SELB  <= '1' when CpldReg6(1) = '0' and MCBSP2_EN = '1' else '0';

FLASH_PAGE  <=     CpldReg6(2);    -- Flash address A19

-- Mapping for CPU Frequency Selection
-- NOTE: ON is logic '0'
--                 SW3-5   SW3-6  SW3-7  SW3-8        Logical
--                 PLLSW1  PLLSW2 PLLSW3 PLLSW4       Binary Value
--    500/100       ON      OFF    OFF     ON         1001
--    600/100       OFF     ON     OFF     ON         1010
--    720/125       ON      ON     OFF     OFF        0011
--    850/125       OFF     OFF    ON      OFF        0100
--    1000/125      OFF     ON     ON      OFF        0110
--    1200/125      ON      ON     ON      OFF        0111
--    1000/125      OFF     OFF    OFF     OFF        0000
--    1000/100      OFF     OFF    OFF     ON         0001
```

```
      CpldReg6(3) <= DSPPLL_SW(0);
      CpldReg6(4) <= DSPPLL_SW(1);
      CpldReg6(5) <= DSPPLL_SW(2);
      CpldReg6(6) <= DSPPLL_SW(3);

-------------------------------------------
-- Mapping ICS521  S1/S0
-------------------------------------------
--    S1      S0       MULTIPLIER
--     0       0           4
--     0       Z           5.333
--     0       1           5
--     Z       0           2.5
--     Z       Z           2
--     Z       1           3.333
--     1       0           6
--     1       Z           3
--     1       1           8
-------------------------------------------
-- Note Logically  Switch On = 0 Off = 1
-------------------------------------------

CPU_DSPPLL_S <=  "ZZ"  when ( DSPPLL_SW = "0110" ) else   --  500 MHz (25*20)
                 "Z0"  when ( DSPPLL_SW = "0101" ) else   --  600 MHz (30*20)
                 "1Z"  when ( DSPPLL_SW = "1100" ) else   --  720 MHz (36*20)
                 "0Z"  when ( DSPPLL_SW = "1011" ) else   --  850 MHz (42.64*20)
                 "00"  when ( DSPPLL_SW = "1001" ) else   -- 1000 MHz (50*20)
                 "01"  when ( DSPPLL_SW = "1000" ) else   -- 1200 MHz (60*20)
                 "00"  ;                                  -- 1000 MHz (50*20)


DSP_PLL_CLK <=  OPT_CLK1_DIV when ( DSPPLL_SW = "0110" ) else   --  12.5 MHz X 2
                CLKIN        when ( DSPPLL_SW = "0101" ) else   --  12.0 MHz X 2.5
                CLKIN        when ( DSPPLL_SW = "1100" ) else   --  12.0 MHz X 3
                OPT_CLK2     when ( DSPPLL_SW = "1011" ) else   --   8.0 MHz X 5.33
                OPT_CLK1_DIV when ( DSPPLL_SW = "1001" ) else   --  12.5 MHz X 4
                CLKIN        when ( DSPPLL_SW = "1000" ) else   --  12.0 MHz X 5
                OPT_CLK1_DIV;

--
-- 6416T MULTIPLIER SET TO 20 TIMES
--
DSP_CLKMODE0 <= '1';
```

```
  DSP_CLKMODE1 <= '1';


   EMIF_PLL_S  <=  "00"  when ( DSPPLL_SW = "0110" ) else   --  500 MHz  100
                   "00"  when ( DSPPLL_SW = "0101" ) else   --  600 MHz  100
                   "01"  when ( DSPPLL_SW = "1100" ) else   --  720 MHz  125
                   "01"  when ( DSPPLL_SW = "1011" ) else   --  850 MHz  125
                   "01"  when ( DSPPLL_SW = "1001" ) else   -- 1000 MHz  125
                   "01"  when ( DSPPLL_SW = "1000" ) else   -- 1200 MHz  125
                   "00"  when ( DSPPLL_SW = "1110" ) else   -- 1000 MHz  100
                   "01"  ;

   EMIF_PLL_CLK <=  OPT_CLK1;                               --  25MHZ

--
--
--

        CpldReg6(7) <= MCBSP2_EN;      -- From PCI header

        -- ============================================================================
        -- Mux the read data from all the registers and output for reads
        --
  process(  DSP_ADDR,CpldReg0,CpldReg1,CpldReg4,CpldReg6 )
        begin
                case DSP_ADDR( 2 downto 0) is
                  when "000"  => MuxD  <= CpldReg0;
                  when "001"  => MuxD  <= CpldReg1;
                  when "100"  => MuxD  <= CpldReg4;
                  when "110"  => MuxD  <= CpldReg6;
                  when others => MuxD  <= "00000000";
               end case;
        end process;

        DSP_DQ <= MuxD  when    DSP_CSn  = '0'
                         and  DSP_REn  = '0'
                         and  DSP_OEn  = '0'
                         else "ZZZZZZZZ";

  -- ############################################################################
  -- Generate the Daughter card buffer control signals. DC buffers are only
  -- enabled if a daughter card is plugged in to mininize EMI.
  --
  -- DSP OE signal is low for read and high for write.  We flip this to match
```

```
  -- the 245 direction control.  The DSP OE signal overlaps the RE/WE signals
  -- so the direction is stable before the enable.
  --
  -- ##############################################################################

  DC_DBUF_DIR <= not DSP_DC_OEn;      -- low for write, high for read

  DC_DBUF_OEn <= '0' when DC_DETn    = '0'
                          and ( DSP_DC_CS0n = '0' or  DSP_DC_CS1n = '0' )
                          and ( DSP_DC_REn  = '0' or  DSP_DC_WEn = '0'  )

                          else '1';

  DC_CNTL_OEn <= '0' when DC_DETn = '0' else '1';

  TEST_PIN <= '0' when (DSPPLL_SW(3)='1') else '1';

end behavior_c6416tdsk;
```