DELAY

COLLEGAMENTI DEL 1° LIVELLO GERARCHICO

delay.vhd è un file che contiene al suo interno, gerarchicamente parlando, tutto l'intero progetto. È sufficiente visionarne il codice per avere un riassunto dei componenti presenti.

CODICE D'INTERCONNESSIONE

```
-- delay.vhd
-- Christian Gregorutti
-- con modifiche di Canziani Alfredo & Viviani Emanuele
-- DEEI, Università degli studi di Trieste

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
-- %                                                                              %
-- %                            LIBRERIE INCLUSE                                  %
-- %                                                                              %
-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use WORK.COMMON.ALL;
use WORK.SDRAM.ALL;

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
-- %                                                                              %
-- %                              PORTE DI DELAY                                  %
-- %                     (terminali di ingresso ed uscita)                        %
-- %                                                                              %
-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

entity delay is
    Generic(
        DATA_WIDTH: natural := 16;          -- SDRAM data width
        BEG_ADDR:   natural := 0;           -- beginning SDRAM address
        END_ADDR:   natural := 4_194_303;   -- ending SDRAM address
        BEG_TEST:   natural := 0;           -- beginning test range address
        END_TEST:   natural := 4_194_303    -- ending test range address
    );
    Port(
        ce_n:   out std_logic;     -- Flash RAM chip-enable
        rstn:   in  std_logic;     -- main system reset input
        clkin:  in  std_logic;     -- main clock input from external clock source
        sclkfb: in  std_logic;     -- feedback SDRAM clock with PCB delays
        s_clk:  out std_logic;     -- clock to SDRAM
        cke:    out std_logic;     -- SDRAM clock-enable
        cs_n:   out std_logic;     -- SDRAM chip-select
        ras_n:  out std_logic;     -- SDRAM RAS
        cas_n:  out std_logic;     -- SDRAM CAS
        we_n:   out std_logic;     -- SDRAM write-enable
        ba:     out unsigned( 1 downto 0);                 -- SDRAM bank-address
        sAddr:  out unsigned(11 downto 0);                 -- SDRAM address bus
        sData:  inout unsigned(DATA_WIDTH-1 downto 0);     -- data bus to SDRAM
        dqmh:   out std_logic;                             -- SDRAM DQMH
        dqml:   out std_logic;                             -- SDRAM DQML

        -- Audio
        lrck:   buffer std_logic;
        mclk:   out std_logic;
```

```
        sclk:   buffer std_logic;
        DataFromAk45120aSerial: in  std_logic;
        DataToAk45120aSerial:   out std_logic;

        -- Keyboard
        --init: in STD_LOGIC;                       -- ps2
        ps2data: inout std_logic;                   -- ps2
        ps2clk:  inout std_logic;                   -- ps2
        display: out std_logic_vector (6 downto 0); -- ps2

        -- switch
        switch : in std_logic_vector (1 downto 0);
        sw_volume : in std_logic_vector(1 downto 0);

        -- LED
        led : out std_logic_vector(6 downto 0)
    );
end delay;

architecture arch of delay is

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
-- %                                                                             %
-- %                                SIGNALS                                      %
-- %                                                                             %
-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

constant ADDR_WIDTH: natural := log2(END_ADDR-BEG_ADDR+1);
signal rst_int: std_logic;                -- internal reset signal
signal clk_int: std_logic;                -- internal master clock signal
signal bufclkin: std_logic;               -- buffered input (non-DLL) clock
signal lock: std_logic;                   -- SDRAM clock DLL lock indicator
signal done: std_logic;                   -- SDRAM operation complete indicator
signal hAddr: unsigned(ADDR_WIDTH-1 downto 0);        -- host address bus
signal hDIn: unsigned(DATA_WIDTH-1 downto 0);         -- host-side data to SDRAM
signal hDOut: unsigned(DATA_WIDTH-1 downto 0);        -- host-side data from SDRAM
signal rd: std_logic;                           -- host-side read control signal
signal wr: std_logic;                           -- host-side write control signal

signal goRegisters: std_logic; -- Sincronizza i registri Seriale/Parallelo e…
    …Parallelo/Seriale
signal latchPiso: std_logic;   -- Dice quando bufferizzare il dato all'interno del…
    …registro dello shift
signal timeDelay: unsigned (16 downto 0);   -- indica il temppo del delay in millisecondi

signal dataFromSdram: unsigned(15 downto 0);
signal dataToSdram: unsigned(15 downto 0);
signal dataToSdramLow: unsigned(15 downto 0);
signal dataFromMix: unsigned(15 downto 0);
signal dataFromSipo: unsigned(15 downto 0);

signal addressRead: unsigned(ADDR_WIDTH-1 downto 0);
signal addressWrite: unsigned(ADDR_WIDTH-1 downto 0);

signal goWR: std_logic;
signal goAddSignal: std_logic;

-- KEYBOARD
signal clk1MHz: std_logic;
signal init: std_logic;

signal switch1: std_logic_vector (1 downto 0);

signal s_sw_volume : std_logic_vector(1 downto 0);
signal s_led : std_logic_vector(6 downto 0);
signal dataToCodecAudio: unsigned(15 downto 0);
```

```
-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
-- %                                                                              %
-- %                              COMPONENTS                                      %
-- %                                                                              %
-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

----------------------------------------------------------------------------------
-- AUDIO
----------------------------------------------------------------------------------

component divisorclk
    Port (   clk:  in     std_logic;
             rstn: in     std_logic;
             lrck: buffer std_logic;
             mclk: out    std_logic;
             sclk: buffer std_logic
             );
end component;

component counterSclk
    Port (   rstn:        in  std_logic;
             sclk_int:    in  std_logic;
             latchPiso:   out std_logic;
             goRegisters: out std_logic;
             goAddSignal: out std_logic;
             goWR:        out std_logic
             );
end component;

component addressGenerator
    Port (   clk:          in  std_logic;
             rstn:         in  std_logic;
             nextAddress:  in  std_logic;
             addressWrite: out unsigned (21 downto 0);
             addressRead:  out unsigned (21 downto 0);
             timeDelay:    in  unsigned (16 downto 0)
             );
end component;

-- Convertotore Seriale/Parallelo
component SIPOshiftreg
    Port (   rstn:  in  std_logic;
             SI:    in  std_logic;
             PO:    out unsigned(15 downto 0);
             Shift: in  std_logic;
             clk:   in  std_logic
             );
end component;

-- Convertitore Parallelo/Seriale
component shiftregPISO
    Port (   dataIn:    in  unsigned (15 downto 0);
             clk:       in  std_logic;
             rstn:      in  std_logic;
             serialout: out std_logic;
             shift:     in  std_logic;
             latch:     in  std_logic
             );
end component;

component mix
    Port (   sclk :          in  std_logic;
             timeDelay :     in  unsigned(16 downto 0);
             goAddSignal :   in  std_logic;
             dataFromSipo :  in  unsigned(15 downto 0);
             dataFromSdram : in  unsigned(15 downto 0);
             dataOut:        out unsigned(15 downto 0);
             switch :        in  std_logic_vector (1 downto 0)
             );
```

```
end component;

-------------------------------------------------------------------------------
-- KEYBOARD
-------------------------------------------------------------------------------

component clock50to1
    Port (   clk50MHz: in  std_logic;
             rstn:     in  std_logic;
             clk1MHz:  out std_logic
             );
end component;

component ps2keyboard
    Port (   clk:       in  std_logic;              -- clock
             rstn:      in  std_logic;              -- asynchronous active low reset
             init:      in  std_logic;              -- inizializza la tastiera
             ps2data:   inout std_logic;            -- bidirectional ps2 data line
             ps2clk:    inout std_logic;            -- bidirectional ps2 clock line
             output:    out std_logic_vector (6 downto 0); -- 7-segment display
             busy:      out std_logic;                     -- Shows when decoder is busy
             timeDelay: out unsigned (16 downto 0)
             );
end component;

-------------------------------------------------------------------------------
-- SDRAM
-------------------------------------------------------------------------------

component WriteRead
    Port (   rstn: in std_logic;
             clk: in std_logic;
             goWR: in std_logic;
             done: in std_logic;
-- Dati che devono essere memorizzati nella SDRAM:
             hDIn: out unsigned(15 downto 0);
-- vengono prima messi in questo vettore
             dataToSdram: in unsigned(15 downto 0);
             hDOut: in unsigned (15 downto 0);
-- Dati che escono dalla SDRAM: vengono bufferizzati in
             dataFromSdram: out unsigned (15 downto 0);
             addressRead: in unsigned(21 downto 0);        -- indirizzi per la lettura
             addressWrite: in unsigned(21 downto 0);       -- indirizzi per la scrittura
             hAddr: out unsigned(21 downto 0);      -- indirizzi che vanno alla SDRAM
             goWrite: out std_logic;
             goRead: out std_logic
             );
end component;

component initialize
    Port (   rstn: in std_logic;
             clk: in std_logic;
             addressWrite: in unsigned(21 downto 0);
             goInitKeyboard: out std_logic
             );
end component;

-------------------------------------------------------------------------------
-- ACCESSORI
-------------------------------------------------------------------------------

component display_alpha
    Port (   sw_alpha : in std_logic_vector(1 downto 0);
             led : out std_logic_vector(6 downto 0)
             );
end component;
```

```
component volume
    Port (   switch_vol : in std_logic_vector(1 downto 0);
             audio_in : in unsigned(15 downto 0);
             audio_out : out unsigned(15 downto 0)
             );
end component;

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
-- %                                                                            %
-- %                                  BEGIN                                     %
-- %                                                                            %
-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

begin

-------------------------------------------------------------------------------------
-- DISABILITO LA SFLASH

    ce_n <= '1'; -- serve per disabilitare la Flash RAM
                     -- in questo caso le linee di I/O connesse alla memoria Flash
                     -- possono essere usate come linee di comunicazione general-purpose
                     -- tra l'FPGA e la CPLD.

-------------------------------------------------------------------------------------

    -- synchronous reset.  internal reset flag is set active by config. bitstream
    -- and then gets reset after DLL clocks start.
    process(bufclkin)
    begin
        if(bufclkin'event and bufclkin='1') then
            if lock='0' then
                rst_int <= '1';        -- keep in reset until DLLs start up
            else
                rst_int <= not rstn;   -- else manually activate reset with pushbutton
            end if;
        end if;
    end process;

-------------------------------------------------------------------------------------

    -- clock 50MHz to 1MHz
    clkadapter: clock50to1 PORT MAP (
    clk50MHz => bufclkin,
    clk1MHz => clk1MHz,
    rstn => rstn
    );

    -- controllore keyboard
    keyboard: ps2keyboard PORT MAP (   clk => clk1MHz,
                                            rstn => rstn,
                                            init => init,
                                            ps2data => ps2data,
                                            ps2clk => ps2clk,
                                            output => display,
                                            busy => open,
                                            timeDelay => timeDelay
                                            );

    -- SDRAM memory controller module
    sdramController: sdramCntl
    generic map (FREQ => 50_000,        -- 50 MHz operation
                     DATA_WIDTH => hDIn'length,
                     NROWS => 4096,
                     NCOLS => 256,
                     HADDR_WIDTH => hAddr'length,
                     SADDR_WIDTH => sAddr'length
                     )
    port map (
        clkin => clkin,            -- master clock from external clock source (unbuffered)
```

```
        bufclk => bufclkin,         -- buffered master clock output
        clk0 => clk_int,            -- synchronized master clock (accounts for delays to…
            …external SDRAM)
        clk2x => open,              -- synchronized doubled master clock
        lock => lock,               -- valid synchronized clocks indicator
        rst => rst_int,             -- reset
        rd => rd,                   -- host-side SDRAM read control from memory tester
        wr => wr,                   -- host-side SDRAM write control from memory tester
        done => done,               -- SDRAM memory read/write done indicator
        hAddr => hAddr,             -- host-side address from memory tester
        hDIn => hDIn,               -- ...
        hDOut => hDOut,             -- SDRAM data output to memory tester
        sdramCntl_state => open,    -- SDRAM controller state (for diagnostics)
        sclkfb => sclkfb,           -- clock feedback with added external PCB delays
        s_clk => s_clk,             -- synchronized clock to external SDRAM
        cke => cke,                 -- SDRAM clock enable
        cs_n => cs_n,               -- SDRAM chip-select
        ras_n => ras_n,             -- SDRAM RAS
        cas_n => cas_n,             -- SDRAM CAS
        we_n => we_n,               -- SDRAM write-enable
        ba => ba,                   -- SDRAM bank address
        sAddr => sAddr,             -- SDRAM address
        sData => sData,             -- SDRAM databus
        dqmh => dqmh,               -- SDRAM DQMH
        dqml => dqml                -- SDRAM DQML
        );

-- Modulo divisorclock
    divisorclk_module: divisorclk PORT MAP(
        clk => bufclkin,
        rstn => rstn,     -- not rst_int, TEST!!!
        lrck => lrck,
        mclk => mclk,
        sclk => sclk
     );

-- Modulo counterSclk
    counterSclk_module: counterSclk PORT MAP(
        rstn => rstn,
        sclk_int => sclk,
        latchPiso => latchPiso,
        goRegisters => goRegisters,
        goAddSignal => goAddSignal,
        goWR => goWR
    );

-- Modulo che genera gli indirizzi
    addressGenerator_module: addressGenerator PORT MAP(
        clk => sclk,
        rstn => rstn,
        nextAddress => latchPiso,
        addressWrite => addressWrite,
        addressRead => addressRead,
        timeDelay => timeDelay
        );

-- Modulo ShiftRegSipo
    SIPOshiftreg_module: SIPOshiftreg PORT MAP(
        rstn => rstn,
        SI => DataFromAk45120aSerial,
        PO => dataFromSipo,
        Shift => goRegisters,
        clk => not sclk
        );
```

```
-- Modulo ShiftRegPiso
     shiftregPISO_module: shiftregPISO PORT MAP(
          dataIn => dataToCodecAudio,
          clk => not sclk,
          rstn => rstn,
          serialout => DataToAk45120aSerial,
          shift => goRegisters,
          latch => latchPiso
          );

-- Modulo Write/Read
     WR_module: WriteRead PORT MAP(
          rstn => rstn,     -- ACHTUNG
          clk => bufclkin,
          goWR => goWR,
          done => done,
          hDIn => hDIn,
          dataToSdram => dataToSdramLow,
          hDOut => hDOut,
          dataFromSdram => dataFromSdram,
          addressRead => addressRead,
          addressWrite => addressWrite,
          hAddr => hAddr,
          goWrite => wr,
          goRead => rd
          );

     mix_module: mix PORT MAP(
          sclk => sclk,
          timeDelay => timeDelay,
          goAddSignal => goAddSignal,
          dataFromSipo => dataFromSipo,
          dataFromSdram => dataFromSdram,
          dataOut => dataFromMix,
          switch => switch1
          );

     initialize_module: initialize PORT MAP(
          rstn => rst_int,
          clk => bufclkin,
          addressWrite => addressWrite,
          goInitKeyboard => init
          );

     display_module: display_alpha PORT MAP (
          switch1,
          s_led
          );

     volume_module: volume PORT MAP (
          s_sw_volume,
          dataFromMix,
          dataToCodecAudio
          );

dataToSdramLow <= dataToSdram;
switch1 <= switch;
s_sw_volume <= sw_volume;
led <= s_led;

dataToSdram_assignment:
dataToSdram <=    dataFromSipo when timeDelay = 0 else
                  dataFromMix;

end arch;
```