library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity controller is

  port(

    btn : in  std\_logic\_vector(3 downto 0);

    sw : in  std\_logic\_vector(0 downto 0);

    o\_mins, o\_secs, o\_wmins : out integer range 0 to 59;

    o\_hours, o\_whours : out integer range 0 to 23;

    alarm : out std\_logic;

    state : out std\_logic\_vector(1 downto 0);

    clk : in  std\_logic;

    reset : in  std\_logic

  );

end controller;

architecture Behavioral of controller is

  component trigger\_gen is

    generic(Delta : integer);

    port(

      clk   : in std\_logic;

      reset : in std\_logic;

      trigger : out std\_logic

    );

  end component trigger\_gen;

  -- pro Viertelsekunde einen Takt lang '1'

  signal fasttrigger : std\_logic;

  -- fasttrigger wird zurueckgesetzt, wenn BTN2 oder BTN3 gedrueckt werden

  signal fasttimer\_reset : std\_logic;

  -- pro Sekunde einen Takt lang '1'

  signal sectrigger : std\_logic;

  signal hours, whours     : integer range 0 to 23;

  signal secs, mins, wmins : integer range 0 to 59;

  -- fuer Flankenerkennung der Taster

  signal btn\_old : std\_logic\_vector(3 downto 0);

  signal btn\_triggered : std\_logic\_vector(3 downto 0);

  type state\_type is (NTIME, SET\_TIME, SET\_ALARM, error);

  signal current\_state : state\_type;

begin

  fasttimer\_reset <= reset or

        btn\_triggered(2) or btn\_triggered(3);

  fasttimer : trigger\_gen

    generic map(11)

    port map(clk, fasttimer\_reset, fasttrigger);

  sectimer : trigger\_gen

    generic map(13)

    port map(clk, reset, sectrigger);

  trigger : process(clk, btn)

  begin

    for i in 0 to 3 loop

      btn\_triggered(i) <= not btn\_old(i) and btn(i);

    end loop;

  end process;

  FSM : process(clk, reset)

    variable next\_state : state\_type;

  begin

    if reset = '1' then

      hours  <= 0;

      mins   <= 0;

      secs   <= 0;

      whours <= 0;

      wmins  <= 0;

      alarm  <= '0';

      current\_state <= NTIME;

    elsif clk'event and clk = '1' then

      for i in 0 to 3 loop

        btn\_old(i) <= btn(i);

      end loop;

      -- Eine Sekunde hochzählen

      secs <= (secs + 1) mod 60;

      -- Wenn Sekunden > 60 + 1 Minute.

      IF secs = 0 THEN

        mins <= (mins + 1) mod 60;

        -- Wenn Minuten > 60 + 1 Stunde

        IF mins = 0 THEN

          hours <= (hours + 1) mod 24;

        END IF;

      END IF;

      -- Alarm ausgeben

      IF  (mins = wmins) AND (hours = whours) THEN

          alarm <= '1';

      ELSE

         -- Alarm deaktivieren

         alarm <= '0';

      END IF;

      case current\_state is

        -- Zustand Time

        when NTIME =>

          -- BTN0' & !BTN1'

          IF (btn\_triggered(0)='1') AND (NOT btn\_triggered(1)='1') THEN

             next\_state := SET\_TIME;

          -- !BTN0' & BTN1'

          ELSIF (NOT btn\_triggered(0)='1') AND btn\_triggered(1)='1' THEN

             next\_state := SET\_ALARM;

          END IF;

        -- Zustand SetTime

        when SET\_TIME =>

          -- BTN0' & !BTN1'

          IF (btn\_triggered(0)='1') AND (NOT btn\_triggered(1)='1') THEN

               next\_state := NTIME;

          END IF;

          -- BTN1' & !BTN0'

          IF (btn\_triggered(1)='1') AND (NOT btn\_triggered(0)='1') THEN

               next\_state := SET\_ALARM;

          END IF;

          -- Minuten nach oben zaehlen BTN2

          IF (btn\_triggered(2)='1') THEN

            mins <= (mins + 1) mod 60;

          END IF;

          -- Stunden nach oben zaehlen BTN3

          IF (btn\_triggered(3)='1') THEN

            hours <= (hours + 1) mod 24;

          END IF;

        -- Zustand SetAlarm

        when SET\_ALARM =>

          -- BTN1' & !BTN0'

          IF (btn\_triggered(1)='1') AND (NOT btn\_triggered(0)='1') THEN

            next\_state := NTIME;

          END IF;

          -- BTN0' & !BTN1'

          IF (btn\_triggered(0)='1') AND (NOT btn\_triggered(1)='1') THEN

            next\_state := SET\_TIME;

          END IF;

          -- Minuten nach oben zaehlen BTN2

          IF (btn\_triggered(2)='1') THEN

            wmins <= (wmins + 1) mod 60;

          END IF;

          -- Stunden nach oben zaehlen BTN3

          IF (btn\_triggered(3)='1') THEN

            whours <= (whours + 1) mod 24;

          END IF;

          -- Illegale Zustaende

        when others =>

          next\_state := NTIME;

      end case;

      current\_state <= next\_state;

    end if;

  end process FSM;

  o\_hours  <= hours;

  o\_mins   <= mins;

  o\_secs   <= secs;

  o\_whours <= whours;

  o\_wmins  <= wmins;

  with current\_state select

    state <= "00" when NTIME,

             "01" when SET\_TIME,

             "10" when SET\_ALARM,

             "11" when others;

end architecture Behavioral;