```
----------------------------------------------------------------------------------
-- Company:  Student
-- Engineer: Emiliano A. Veiga - veigaemiliano@gmail.com
--
-- Create Date:    17:37:41 07/12/2011
-- Design Name:
-- Module Name:    my_dds - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
----------------------------------------------------------------------------------
ENTITY my_dds IS

    PORT(
         enable: IN  std_logic;
          clock: IN  std_logic;
           sin: OUT std_logic_vector(7 downto 0);
          reset: IN  std_logic;
         led_on: OUT std_logic;
          btn_0: IN std_logic;
          switch: IN  std_logic_vector(7 downto 0)
        );

END my_dds;
----------------------------------------------------------------------------------
ARCHITECTURE Behavioral OF my_dds IS

--DDS component
COMPONENT dds_component
    PORT (
    ce: IN std_logic;
    clk: IN std_logic;
    pinc_in: IN std_logic_vector(7 downto 0);
    sine: OUT std_logic_vector(7 downto 0));
END COMPONENT;

--DCM component
COMPONENT dcm_hardware
    PORT(
        CLKIN_IN : IN std_logic;
        RST_IN : IN std_logic;
        CLKFX_OUT : OUT std_logic;
        CLKIN_IBUFG_OUT : OUT std_logic;
        CLK0_OUT : OUT std_logic;
        CLK2X_OUT : OUT std_logic
        );
    END COMPONENT;
```

```
--DDS signals
attribute syn_black_box : boolean;
attribute syn_black_box of dds_component: component is true;
signal wave         : std_logic_vector(7 downto 0);
signal wave_inv     : std_logic_vector(7 downto 0);
signal phase_inc    : std_logic_vector(7 downto 0);

--sinais para geraçao de outro clock
signal clk_new : std_logic;
signal counter:integer;
signal modulo:integer;

signal clk_botao : std_logic;
signal btn_count : integer;
signal step:integer;

signal clk_system:std_logic; --50Mhz board source
signal clk_2x: std_logic;    --100Mhz ouput from DCM

BEGIN

    gerador: dds_component --DDS 4.0
        port map (
            ce => enable,
            clk => clk_new,
            pinc_in => phase_inc,
            sine => wave
            );

    my_dcm: dcm_hardware PORT MAP( --DCM Director Clock Manager
        CLKIN_IN => clock,
        RST_IN => reset,
        CLKFX_OUT => open,
        CLKIN_IBUFG_OUT => open,
        CLK0_OUT => clk_system,
        CLK2X_OUT => clk_2x
    );

    process(clk_system) --clock button
        begin
            if(rising_edge(clk_system)) then
                if(btn_count = 3000100) then --15kHz button event
                    clk_botao <= not clk_botao;
                    btn_count <= 0;
                  else
                    btn_count <= btn_count + 1;
                end if;
            end if;
    end process;

    process(clk_2x,enable) --clock divisor for DDS
        begin
            if(clk_2x ='1' and clk_2x'EVENT) then
               if(enable = '1') then
                  if(counter = modulo) then
                      clk_new <= not clk_new;
```

```
                    counter <= 0;
                    else counter <= counter + 1;
                end if;
                    else counter <= 0; clk_new <= '0';
            end if;
        end if;
    end process;


    process(btn_0,clk_botao)--counter
        begin
            if(clk_botao = '1' and clk_botao'EVENT) then
                if(btn_0 = '1') then
                    step <= step + 1;
                        if(step = 11) then
                            step <= 0;
                        end if;
                end if;
            end if;
    end process;



  process(step,clk_botao)
    begin               --clock divisor counter
      if(rising_edge(clk_botao)) then

        case step is
                when 1 =>
                        modulo <= 651;
                when 2 =>
                        modulo <= 195;
                when 3 =>
                        modulo <= 38;
                when 4 =>
                        modulo <= 23;
                when 5 =>
                        modulo <= 18;
                when 6 =>
                        modulo <= 12;
                when 7 =>
                        modulo <= 4;
                when 8 =>
                        modulo <= 3;
                when 9 =>
                        modulo <= 2;
                when 10 =>
                        modulo <= 1;
                when others =>
                        modulo <= 3900;
            end case;
       end if;
  end process;

    --phase increment (switch change in runtime by user)
    phase_inc <= switch;

    --revert bit 7 for compatibility with DAC0800
```

```
        wave_inv(6 downto 0) <= wave(6 downto 0);
        wave_inv(7) <= not wave(7);

        --digital signal output (8bit)
        sin <= wave_inv;

        --indicate that generator is turned on
        led_on <= enable;


END Behavioral;
```