```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity moduldac_wrap is


port (
      o_clock_dac     : out  std_logic := '1';
      o_data_dac      : out  std_logic := '0';
      o_sync_dac      : out  std_logic := '1';
      o_ldac_dac      : out  std_logic := '1';
      o_clr_dac       : out  std_logic := '1';
      i_clk           : in   std_logic := '0';
      i_rst           : in   std_logic := '0';
      i_update        : in   std_logic := '0';
      i_data_att1_dac: in std_logic_vector(11 downto 0) := (others=> '0');
      i_data_att2_dac: in std_logic_vector(11 downto 0) := (others=> '0');
      i_data_maam_i   : in std_logic_vector(11 downto 0) := (others=> '0');
      i_data_maam_g   : in std_logic_vector(11 downto 0) := (others=> '0');
      i_channel       : in std_logic_vector(2 downto 0) := (others=> '0')


);

end moduldac_wrap;

architecture beexam of moduldac_wrap is

type State_type is (init_outrange,outrange_wait,init_powcont,powcont_wait,init_powcont_2,
powcont_wait_2,init_dac,init_cont,start_update,activ_signals);
   signal State : State_Type;

signal start,data_dac,enable,done,mask_ldac,mask_update,update_all,init : std_logic := '0
';
signal sync_dac,clock_dac,ldac                                          : std_logic := '1
';
signal data, trn_data,data_A,data_B,data_C,data_D                       :
std_logic_vector(23 downto 0) := (others=> '0');
signal counter                                                          : integer range 0
 to 1200 := 0;
signal wrt                                                              : integer range 0
 to 24 := 0;
signal num_ch                                                           : integer range 0
 to 3 := 0;
signal c_data                                                           : unsigned(4
downto 0) := (others=> '0');
signal tx                                                               : unsigned(1
downto 0) := (others=> '0');

begin


   process (i_clk) begin                       ------------управление sync, counter, enable

      if rising_edge(i_clk) then

         if (i_rst='1') then
            counter <= 0;

         else
            --по start поднимаем enablee
            --через 24 спада опускаем enable
            if (counter = 105) then                --24 спада
               enable <='0';
            elsif (start ='1') then
               enable <='1';
            end if;

            --считаем такты, когда done нуль
            if (done='0') then
               counter <= counter + 1;
            elsif (done = '1') then
               counter <= 0;
            end if;

            --управление sync
            if (enable = '1')  then
                  if (counter = 10) then
                     sync_dac <= '0';
                  end if;
            else
               sync_dac <= '1';
            end if;

         end if;
      end if;
   end process;


   process (i_clk) begin                        ------------управление clock_dac

      if rising_edge(i_clk) then
         if (i_rst='1') then
            clock_dac <= '1';
         else
            if (sync_dac = '1') then
               clock_dac <= '1';
            elsif (counter rem 2 = 0) and (counter >= 12) then
               clock_dac <= NOT(clock_dac);
            end if;
         end if;
      end if;
   end process;


   process (i_clk) begin                        ------------write bites

      if rising_edge(i_clk) then
         if (i_rst='1') then
            wrt <= 0;
            done <= '0';
            num_ch <= 0;

         else
            if (start='1') then
               done<='0';
            end if;
            --в trn_data записываем биты
            if (enable = '1') and (counter=2) then
               if (update_all = '0') then
                  trn_data <= data;
               else
                  if (num_ch=0) then
                     trn_data <= data_A;
                  elsif (num_ch=1) then
                     trn_data <= data_B;
                  elsif (num_ch=2) then
                     trn_data <= data_C;
                  elsif (num_ch=3) then
                     trn_data <= data_D;
                  end if;
               end if;
            end if;

            --управление done, обнуление wrt
            if (update_all='1') and (counter=108) and (num_ch<3) then
               num_ch <= num_ch + 1;
               done<='1';
               wrt <= 0;
            else
               if (counter=1124) and (mask_ldac='1') then
                  done<='1';
                  wrt <= 0;
               elsif (counter=1110) and (mask_ldac='0') then
                  done<='1';
                  wrt <= 0;
               end if;
            end if;

            --в data_dac передаем по 1 биту
            if (enable='1') and (counter=10+wrt*4) then
               wrt <= wrt + 1;
               data_dac <= trn_data(23);                          --первый передаваемый
бит = старший бит
               trn_data <= trn_data(22 downto 0) & trn_data(23);
            end if;


            if (update_all='0') then
               num_ch <= 0;
            end if;

         end if;
      end if;
   end process;


   process (i_clk) begin                                  ------------state machine

      if rising_edge(i_clk) then
         if (i_rst='1') then
            State <= init_outrange;
            ldac <= '1';
            start <= '0';
            mask_ldac <= '1';

         else

            case State is

               --инициализация
               when init_outrange =>
                  if (start='1') then
                     start <= '0';
                     state <= outrange_wait;
                  else
                     start <= '1';
                     data <= "000011000000000000000011" ; --for Output range select register
                  end if;

               when outrange_wait =>
                  if (done='1')  then
                     State <= init_powcont;
                  end if;

               when init_powcont =>

                  if (start='1') then
                     start <= '0';
                     state <= powcont_wait;
                  else
                     start <= '1';
                     data <= "000100000000000000001111" ; --for power control register
                  end if;

               when powcont_wait =>
                  if (done='1') then
                     State <= init_powcont_2;
                  end if;

               when init_powcont_2 =>

                  if (start='1') then
                     start <= '0';
                     state <= powcont_wait_2;
                  else
                     start <= '1';
                     data <= "000100000000000000001111" ; --for power control register
                  end if;

               when powcont_wait_2 =>
                  if (done='1') then
                     State <= init_dac;
                  end if;

               --заружаем значения по умолчанию в DAC A,B,C,D
               when init_dac =>

                  if (start='1') then
                     update_all <= '1';
                     start <= '0';
                     init <= '1';
                     State <= activ_signals;
                  else
                     mask_ldac <= '1';
                     start <= '1';
                     data_A <= "000000001001100110100000" ; --for DAC A register
                     data_B <= "000000011001100110100000" ; --for DAC B register
                     data_C <= "000000100101011100010000" ; --for DAC C register
                     data_D <= "000000110110011001100000" ; --for DAC D register
```

```vhdl
                        end if;

                    --управление start при инициализации
                    when init_cont =>

                        if (start='1') then
                            start <= '0';
                            State <= activ_signals;
                        else
                            start <= '1';
                        end if;
                    --конец инициализации

                    --обновление
                    when start_update =>
                        if (i_update='1') then
                            mask_update<='1';

                            case i_channel is

                                    when "000" =>
                                        data <= "00000" & i_channel & i_data_att1_dac & "0000" ; --for DAC A register

                                    when "001" =>
                                        data <= "00000" & i_channel & i_data_att2_dac & "0000" ; --for DAC B register

                                    when "010" =>
                                        data <= "00000" & i_channel & i_data_maam_i & "0000" ; --for DAC C register

                                    when "011" =>
                                        data <= "00000" & i_channel & i_data_maam_g & "0000"; --for DAC D register

                                    when "100" =>
                                        update_all <= '1';
                                        data_A <= "00000000" & i_data_att1_dac & "0000";   --for DAC A register
                                        data_B <= "00000001" & i_data_att2_dac & "0000" ;  --for DAC B register
                                        data_C <= "00000010" & i_data_maam_i & "0000" ;    --for DAC C register
                                        data_D <= "00000011" & i_data_maam_g & "0000";     --for DAC D register

                                    when others =>
                                        data <= (others=> '0');
                                end case;
                        end if;

                        if (mask_update='1') then
                            if (start='1') then
                                start <= '0';
                                State <= activ_signals;
                            else
                                start <= '1';
                                mask_ldac <= '1';
                            end if;
                        end if;

                    -- управление ldac, init, mask_ldac
                    when activ_signals =>

                        if (num_ch=3) or (update_all='0') then
                            if (counter=120) then       --формирование стрба ldac
                                ldac <= '0';
                            elsif (counter=123) then
                                ldac <= '1';
                            elsif (counter=1124) then
                                mask_ldac<='0';
                                update_all <= '0';
                                init <= '0';
                            end if;
                        end if;

                        if (done='1') then
                            if (update_all = '0') then
                                State <= start_update;
                                mask_update<='0';
                        elsif (init='1') then
                           State <= init_cont;
                        else
                           State <= start_update;
                        end if;
                     end if;
               end case;
            end if;
         end if;
      end process;

   o_ldac_dac <= ldac;
   o_clock_dac <= clock_dac;
   o_sync_dac <= sync_dac;
   o_data_dac <= data_dac;
   end beexam;
```