

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5
6  entity moduldac_wrap is
7
8
9  port (
10     o_clock_dac      : out  std_logic := '1';
11     o_data_dac       : out  std_logic := '0';
12     o_sync_dac       : out  std_logic := '1';
13     o_ldac_dac       : out  std_logic := '1';
14     o_clr_dac        : out  std_logic := '1';
15     i_clk            : in   std_logic := '0';
16     i_rst            : in   std_logic := '0';
17     i_update         : in   std_logic := '0';
18     i_data_att1_dac  : in   std_logic_vector(11 downto 0) := (others=> '0');
19     i_data_att2_dac  : in   std_logic_vector(11 downto 0) := (others=> '0');
20     i_data_maam_i    : in   std_logic_vector(11 downto 0) := (others=> '0');
21     i_data_maam_g    : in   std_logic_vector(11 downto 0) := (others=> '0');
22     i_channel        : in   std_logic_vector(2  downto 0) := (others=> '0');
23
24
25 );
26
27 end moduldac_wrap;
28
29 architecture beexam of moduldac_wrap is
30
31     type State_type is (init_outrange,outrange_wait,init_powcont,powcont_wait,init_powcont_2 ,
32     powcont_wait_2,init_dac,init_cont,start_update,activ_signals);
33     signal State : State_Type;
34
35     signal start,data_dac,enable,done,mask_ldac,mask_update,update_all,init : std_logic := '0';
36
37     signal sync_dac,clock_dac,ldac : std_logic := '1';
38
39     signal data, trn_data,data_A,data_B,data_C,data_D :
40     std_logic_vector(23 downto 0) := (others=> '0');
41
42     signal counter : integer range 0
43     to 1200 := 0;
44
45     signal wrt : integer range 0
46     to 24 := 0;
47
48     signal num_ch : integer range 0
49     to 3 := 0;
50
51     signal c_data : unsigned(4
52     downto 0) := (others=> '0');
53
54     signal tx : unsigned(1
55     downto 0) := (others=> '0');
56
57 begin
58
59     process (i_clk) begin
60         -----управление sync, counter, enable
61
62         if rising_edge(i_clk) then
63
64             if (i_rst='1') then
65                 counter <= 0;
66
67             else
68                 --по start поднимаем enablee
69                 --через 24 спада опускаем enable
70                 if (counter = 105) then
71                     enable <='0';
72                     --24 спада
73                 elsif (start='1') then
74                     enable <='1';
75                 end if;
76
77                 --считаем такты, когда done нуль
78                 if (done='0') then
79                     counter <= counter + 1;
80                 elsif (done = '1') then
81                     counter <= 0;
82                 end if;
83
84                 --управление sync
85                 if (enable = '1') then
86                     if (counter = 10) then
87                         sync_dac <= '0';
88                     end if;
89                 end if;
90             end if;
91         end process;
92
93     end architecture beexam;
94
95 
```

```

73         end if;
74     else
75         sync_dac <= '1';
76     end if;
77
78     end if;
79 end if;
80 end process;
81
82
83 process (i_clk) begin                                -----управление clock_dac
84
85     if rising_edge(i_clk) then
86         if (i_rst='1') then
87             clock_dac <= '1';
88         else
89             if (sync_dac = '1') then
90                 clock_dac <= '1';
91             elsif (counter rem 2 = 0) and (counter >= 12) then
92                 clock_dac <= NOT(clock_dac);
93             end if;
94         end if;
95     end if;
96 end process;
97
98
99 process (i_clk) begin                                -----write bites
100
101     if rising_edge(i_clk) then
102         if (i_rst='1') then
103             wrt <= 0;
104             done <= '0';
105             num_ch <= 0;
106
107         else
108             if (start='1') then
109                 done<='0';
110             end if;
111             --в trn_data записываем биты
112             if (enable = '1') and (counter=2) then
113                 if (update_all = '0') then
114                     trn_data <= data;
115                 else
116                     if (num_ch=0) then
117                         trn_data <= data_A;
118                     elsif (num_ch=1) then
119                         trn_data <= data_B;
120                     elsif (num_ch=2) then
121                         trn_data <= data_C;
122                     elsif (num_ch=3) then
123                         trn_data <= data_D;
124                     end if;
125                 end if;
126             end if;
127
128             --управление done, обнуление wrt
129             if (update_all='1') and (counter=108) and (num_ch<3) then
130                 num_ch <= num_ch + 1;
131                 done<='1';
132                 wrt <= 0;
133             else
134                 if (counter=1124) and (mask_ldac='1') then
135                     done<='1';
136                     wrt <= 0;
137                 elsif (counter=1110) and (mask_ldac='0') then
138                     done<='1';
139                     wrt <= 0;
140                 end if;
141             end if;
142
143             --в data_dac передаем по 1 биту
144             if (enable='1') and (counter=10+wrt*4) then
145                 wrt <= wrt + 1;
146                 data_dac <= trn_data(23);                                --первый передаваемый
147                 бит = старший бит
148                 trn_data <= trn_data(22 downto 0) & trn_data(23);
149             end if;
150
151             if (update_all='0') then
152                 num_ch <= 0;

```

```

153         end if;
154
155     end if;
156 end if;
157 end process;
158
159
160 process (i_clk) begin
161                                     -----state machine
162     if rising_edge(i_clk) then
163         if (i_rst='1') then
164             State <= init_outrange;
165             ldac <= '1';
166             start <= '0';
167             mask_ldac <= '1';
168
169         else
170             case State is
171             --инициализация
172             when init_outrange =>
173                 if (start='1') then
174                     start <= '0';
175                     state <= outrange_wait;
176                 else
177                     start <= '1';
178                     data <= "00001100000000000000000011"; --for Output range select
179
180             register
181                 end if;
182
183             when outrange_wait =>
184                 if (done='1') then
185                     State <= init_powcont;
186                 end if;
187
188             when init_powcont =>
189                 if (start='1') then
190                     start <= '0';
191                     state <= powcont_wait;
192                 else
193                     start <= '1';
194                     data <= "0001000000000000000000001111"; --for power control register
195                 end if;
196
197             when powcont_wait =>
198                 if (done='1') then
199                     State <= init_powcont_2;
200                 end if;
201
202             when init_powcont_2 =>
203                 if (start='1') then
204                     start <= '0';
205                     state <= powcont_wait_2;
206                 else
207                     start <= '1';
208                     data <= "0001000000000000000000001111"; --for power control register
209                 end if;
210
211             when powcont_wait_2 =>
212                 if (done='1') then
213                     State <= init_dac;
214                 end if;
215
216             --загружаем значения по умолчанию в DAC A,B,C,D
217             when init_dac =>
218                 if (start='1') then
219                     update_all <= '1';
220                     start <= '0';
221                     init <= '1';
222                     State <= activ_signals;
223                 else
224                     mask_ldac <= '1';
225                     start <= '1';
226                     data_A <= "000000001001100110100000"; --for DAC A register
227                     data_B <= "000000011001100110100000"; --for DAC B register
228                     data_C <= "000000100101011100010000"; --for DAC C register
229                     data_D <= "000000110110011001100000"; --for DAC D register
230
231
232

```

```

233         end if;
234
235     --управление start при инициализации
236     when init_cont =>
237
238         if (start='1') then
239             start <= '0';
240             State <= activ_signals;
241         else
242             start <= '1';
243         end if;
244     --конец инициализации
245
246     --обновление
247     when start_update =>
248         if (i_update='1') then
249             mask_update<='1';
250
251             case i_channel is
252
253                 when "000" =>
254                     data <= "00000" & i_channel & i_data_att1_dac & "0000" ;
255
256                 when "001" =>
257                     data <= "00000" & i_channel & i_data_att2_dac & "0000" ;
258
259                 when "010" =>
260                     data <= "00000" & i_channel & i_data_maam_i & "0000" ;
261
262                 when "011" =>
263                     data <= "00000" & i_channel & i_data_maam_g & "0000";
264
265                 when "100" =>
266                     update_all <= '1';
267                     data_A <= "00000000" & i_data_att1_dac & "0000";    --for
268                                     data_B <= "00000001" & i_data_att2_dac & "0000" ;    --for
269                                     data_C <= "00000010" & i_data_maam_i & "0000" ;    --for
270                                     data_D <= "00000011" & i_data_maam_g & "0000";    --for
271
272                 when others =>
273                     data <= (others=> '0');
274             end case;
275         end if;
276
277         if (mask_update='1') then
278             if (start='1') then
279                 start <= '0';
280                 State <= activ_signals;
281             else
282                 start <= '1';
283                 mask_ldac <= '1';
284             end if;
285         end if;
286
287     -- управление ldac, init, mask_ldac
288     when activ_signals =>
289
290         if (num_ch=3) or (update_all='0') then
291             if (counter=120) then    --формирование стрба ldac
292                 ldac <= '0';
293             elsif (counter=123) then
294                 ldac <= '1';
295             elsif (counter=1124) then
296                 mask_ldac<='0';
297                 update_all <= '0';
298                 init <= '0';
299             end if;
300         end if;
301
302         if (done='1') then
303             if (update_all = '0') then
304                 State <= start_update;
305                 mask_update<='0';

```

```
306         elsif (init='1') then
307             State <= init_cont;
308         else
309             State <= start_update;
310         end if;
311     end if;
312 end case;
313 end if;
314 end if;
315 end process;
316
317 o_ldac_dac <= ldac;
318 o_clock_dac <= clock_dac;
319 o_sync_dac <= sync_dac;
320 o_data_dac <= data_dac;
321 end beexam;
```