```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date:    09:31:24 09/14/2015
7   // Design Name:
8   // Module Name:    spi_top
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21  module spi_top(
22  input clk,              //200MHz
23  input rstn,
24  //input spi_start,
25  output wire spi_csn1_pll,
26  output wire spi_sck,
27  output wire spi_mosi,
28  output wire spi_mosi_test,
29  output wire spi_sck_test,
30  output wire spi_csn1_pll_test,
31  //output reg spi_pll_end,
32  output wire spi_pll_end_led
33  );
34
35  wire spi_start;
36  reg clk_spi;//10 Mhz
37  //reg spi_sck_buffer=0;
38  reg spi_mosi_buffer;
39
40  reg [2:0]current_state;
41  reg [2:0]next_state;
42
43  parameter[2:0] ST_IDLE=3'b000;
44  parameter[2:0] ST_SEND_R=3'b001;
45  parameter[2:0] ST_SEND_C=3'b010;
46  parameter[2:0] ST_SEND_WAIT=3'b011;
47  parameter[2:0] ST_SEND_N=3'b100;
48
49  parameter[23:0] R_COUNTER_DATA=24'h300141;//0011 0000 0000 0000 1100 1001
50  parameter[23:0] CONTROL_DATA=24'h4FF924;//  0100 1111 1111 0001 0000 0100
51  parameter[23:0] N_COUNTER_DATA=24'h30D12;// 0000 0011 0000 1101 0001 0010
52
53  reg[23:0] send_data_buffer;
54  reg[23:0] send_R_buffer;
55  reg[23:0] send_C_buffer;
56  reg[23:0] send_N_buffer;
57
58  reg [7:0] clk_spi_cnt;
59  reg [3:0] spi_start_cnt;
60  reg[4:0] R_cnt;
61  reg[4:0] C_cnt;
62  reg[4:0] N_cnt;
```

```
 63   reg[19:0] delay_cnt;
 64
 65   wire R_end;
 66   wire C_end;
 67   wire N_end;
 68   wire delay_end;
 69
 70   reg spi_pll_end;
 71   ////////////////////////////////////////////////////////////////////////////
 72   always @(posedge clk or  negedge rstn) //分频计数
 73   begin
 74      if(!rstn)
 75         clk_spi_cnt<=8'd0;
 76      else if(clk_spi_cnt==8'd99)
 77         clk_spi_cnt<=8'd0;
 78         else
 79         clk_spi_cnt<=clk_spi_cnt+1;
 80   end
 81
 82   always @(posedge clk or  negedge rstn) //generate 1Mhz spi clock
 83   begin
 84      if(!rstn)
 85         clk_spi<=1'b0;
 86      else if(clk_spi_cnt==8'd0)
 87         clk_spi<=~clk_spi;
 88         else
 89         clk_spi<=clk_spi;
 90   end
 91
 92   always @(posedge clk_spi or  negedge rstn) //generate spi_start
 93   begin
 94      if(!rstn)
 95         spi_start_cnt<=4'd0;
 96      else if(spi_start_cnt==4'd15)
 97         spi_start_cnt<=spi_start_cnt;
 98         else
 99         spi_start_cnt<=spi_start_cnt+1;
100   end
101
102   assign spi_start=(spi_start_cnt>6)&&(spi_start_cnt<15);
103
104
105   ////////////////////////////////////////////////////////////////////////////
106   always @(posedge clk_spi or  negedge rstn)
107   begin
108      if(!rstn)
109         current_state<=ST_IDLE;
110         else
111         current_state<=next_state;
112   end
113
114
115   always @(current_state or spi_start or R_end or C_end or N_end or delay_end)
116   begin
117      next_state=current_state;
118      case(current_state)
119      ST_IDLE: begin
120                  if(spi_start==1'b1)
121                     next_state=ST_SEND_R;
122
123               end
124
```

```
125      ST_SEND_R:  begin
126                     if(R_end==1'b1)
127                        next_state=ST_SEND_C;
128
129                     end
130
131      ST_SEND_C:  begin
132                     if(C_end==1'b1)
133                        next_state=ST_SEND_WAIT;
134
135                  end
136
137      ST_SEND_WAIT:  begin
138                        if(delay_end==1'b1)
139                           next_state=ST_SEND_N;
140
141                     end
142
143      ST_SEND_N:  begin
144                     if(N_end==1'b1)
145                        next_state=ST_IDLE;
146
147                  end
148
149      endcase
150   end
151
152   ////////////////////////////////////////////////////////////////////////////
153   always @(posedge clk_spi or  negedge rstn)
154   begin
155      if(!rstn)
156         R_cnt<=0;
157      else if(R_cnt==5'd25)
158         R_cnt<=0;
159      else if(current_state==ST_SEND_R)
160         R_cnt<=R_cnt+1'd1;
161   end
162
163   always @(posedge clk_spi or  negedge rstn)
164   begin
165      if(!rstn)
166         C_cnt<=0;
167      else if(C_cnt==5'd25)
168         C_cnt<=0;
169      else if(current_state==ST_SEND_C)
170         C_cnt<=C_cnt+1'd1;
171   end
172
173   always @(posedge clk_spi or  negedge rstn)
174   begin
175      if(!rstn)
176         N_cnt<=0;
177      else if(N_cnt==5'd25)
178         N_cnt<=N_cnt;
179      else if(current_state==ST_SEND_N)
180         N_cnt<=N_cnt+1'd1;
181   end
182
183   always @(posedge clk_spi or  negedge rstn)//delay 6ms
184   begin
185      if(!rstn)
186         delay_cnt<=0;
```

```
187      else if(delay_cnt==20'd6000)
188         delay_cnt<=0;
189      else if(current_state==ST_SEND_WAIT)
190         delay_cnt<=delay_cnt+1'd1;
191   end
192   ////////////////////////////////////////////////////////////////////////////
193   always @(negedge   clk_spi or  negedge rstn)
194   begin
195      if(!rstn)
196         send_R_buffer<=R_COUNTER_DATA;
197      else if(current_state==ST_SEND_R)
198         send_R_buffer<=send_R_buffer<<1;
199      else
200         send_R_buffer<=send_R_buffer;
201   end
202
203   always @(negedge clk_spi or  negedge rstn)///change to negedge
204   begin
205      if(!rstn)
206         send_C_buffer<=CONTROL_DATA;
207      else if(current_state==ST_SEND_C)
208         send_C_buffer<=send_C_buffer<<1;
209      else
210         send_C_buffer<=send_C_buffer;
211   end
212
213   always @(negedge clk_spi or  negedge rstn)
214   begin
215      if(!rstn)
216         send_N_buffer<=N_COUNTER_DATA;
217      else if(current_state==ST_SEND_N)
218         send_N_buffer<=send_N_buffer<<1;
219      else
220         send_N_buffer<=send_N_buffer;
221   end
222   ////////////////////////////////////////////////////////////////////////////
223   always @(negedge clk_spi or  negedge rstn)
224   begin
225   if(!rstn)
226      spi_mosi_buffer<=1'b0;
227      case(current_state)
228      ST_SEND_R:  spi_mosi_buffer<=send_R_buffer[23];
229
230      ST_SEND_C:  spi_mosi_buffer<=send_C_buffer[23];
231
232      ST_SEND_N:  spi_mosi_buffer<=send_N_buffer[23];
233
234      default:spi_mosi_buffer<=1'b0;
235
236      endcase
237   end
238   ////////////////////////////////////////////////////////////////////////////
239
240   assign  R_end=(R_cnt==5'd25)&&(send_R_buffer==24'd0);
241   assign  C_end=(C_cnt==5'd25)&&(send_C_buffer==24'd0);
242   assign  delay_end=delay_cnt==20'd6000;
243   assign  N_end=(N_cnt==5'd25)&&(send_N_buffer==24'd0);
244
245   always @(posedge clk_spi or  negedge rstn)
246   begin
247   if(!rstn)
248      spi_pll_end<=0;
```

```
249         else
250         spi_pll_end<=N_end;
251
252   end
253
254   assign  spi_csn1_pll=!((current_state==ST_SEND_R)||(current_state==ST_SEND_C)||(
      current_state==ST_SEND_N))||R_end||C_end||N_end;
255   assign spi_mosi=spi_mosi_buffer;   //output data
256   assign spi_sck=clk_spi;            //output clk
257   assign spi_pll_end_led=(N_cnt==5'd25);
258
259   assign spi_sck_test=clk_spi;
260   assign spi_mosi_test=spi_mosi_buffer;   //output data
261   assign spi_csn1_pll_test=spi_csn1_pll;
262
263   endmodule
264
```