```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:      Markus Johansson
4  --
5  -- Create Date:    20:40:19 11/26/2008
6  -- Design Name:
7  -- Module Name:    SteppermotorController - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity SteppermotorController is
31    Port( STEP  : in  STD_LOGIC;
32          DIR   : in  STD_LOGIC;
33          EN    : in  STD_LOGIC; -- Enable
34          Full_Half_Step : in STD_LOGIC; -- '1' half-step, '0' full-step
35          MOTOR : out std_logic_vector(3 downto 0));
36 end SteppermotorController;
37
38 architecture Behavioral of SteppermotorController is
39    type state_motor_type is (step0,step1,step2,step3,step4,step5,step6,step7);
40    signal state_motor, next_state_motor : state_motor_type;
41
42 begin
43    SYNC_PROC: process (STEP)
44    begin
45       if (STEP'event and STEP = '1') then
46          if (EN = '0') then
47             state_motor <= next_state_motor;
48          end if;
49       end if;
50    end process;
51
52    --MOORE State-Machine - Outputs based on state only
53    NEXT_STATE_MOTOR_DECODE: process (state_motor)
54    begin
55       case (state_motor) is
56
57          when step0 =>
58             if DIR='1' then
59                if Full_Half_Step='1' then
60                   next_state_motor <= step1;
61                else
```

```
62                         next_state_motor <= step2;
63                      end if;
64                   else
65                      if Full_Half_Step='1' then
66                         next_state_motor <= step7;
67                      else
68                         next_state_motor <= step6;
69                      end if;
70                   end if;
71
72                when step1 =>
73                   if DIR='1' then
74                      next_state_motor <= step2;
75                   else
76                      next_state_motor <= step0;
77                   end if;
78
79                when step2 =>
80                   if DIR='1' then
81                      if Full_Half_Step='1' then
82                         next_state_motor <= step3;
83                      else
84                         next_state_motor <= step4;
85                      end if;
86                   else
87                      if Full_Half_Step='1' then
88                         next_state_motor <= step1;
89                      else
90                         next_state_motor <= step0;
91                      end if;
92                   end if;
93
94                when step3 =>
95                   if DIR='1' then
96                      next_state_motor <= step4;
97                   else
98                      next_state_motor <= step2;
99                   end if;
100
101               when step4 =>
102                  if DIR='1' then
103                     if Full_Half_Step='1' then
104                        next_state_motor <= step5;
105                     else
106                        next_state_motor <= step6;
107                     end if;
108                  else
109                     if Full_Half_Step='1' then
110                        next_state_motor <= step3;
111                     else
112                        next_state_motor <= step2;
113                     end if;
114                  end if;
115
116               when step5 =>
117                  if DIR='1' then
118                     next_state_motor <= step6;
119                  else
120                     next_state_motor <= step4;
121                  end if;
122
```

```
123                when step6 =>
124                   if DIR='1' then
125                      if Full_Half_Step='1' then
126                         next_state_motor <= step7;
127                      else
128                         next_state_motor <= step0;
129                      end if;
130                   else
131                      if Full_Half_Step='1' then
132                         next_state_motor <= step5;
133                      else
134                         next_state_motor <= step4;
135                      end if;
136                   end if;
137
138                when step7 =>
139                   if DIR='1' then
140                      next_state_motor <= step0;
141                   else
142                      next_state_motor <= step6;
143                   end if;
144
145                when others =>
146                   next_state_motor <= step0;
147
148             end case;
149          end process;
150
151          -- Stepping Sequences
152          -- #        full/half
153          -- 0 = 1100 x     x
154          -- 1 = 0100       x
155          -- 2 = 0110 x     x
156          -- 3 = 0010       x
157          -- 4 = 0011 x     x
158          -- 5 = 0001       x
159          -- 6 = 1001 x     x
160          -- 7 = 1000       x
161          OUTPUT_DECODE: process (state_motor)
162          begin
163             case (state_motor) is
164
165                when step0 =>
166                   MOTOR <= "1100";
167
168                when step1 =>
169                   MOTOR <= "0100";
170
171                when step2 =>
172                   MOTOR <= "0110";
173
174                when step3 =>
175                   MOTOR <= "0010";
176
177                when step4 =>
178                   MOTOR <= "0011";
179
180                when step5 =>
181                   MOTOR <= "0001";
182
183                when step6 =>
```

```
184                     MOTOR <= "1001";
185
186                 when step7 =>
187                     MOTOR <= "1000";
188
189                 when others =>
190                     MOTOR <= "1100";
191
192             end case;
193         end process;
194
195     end Behavioral;
196
197
```