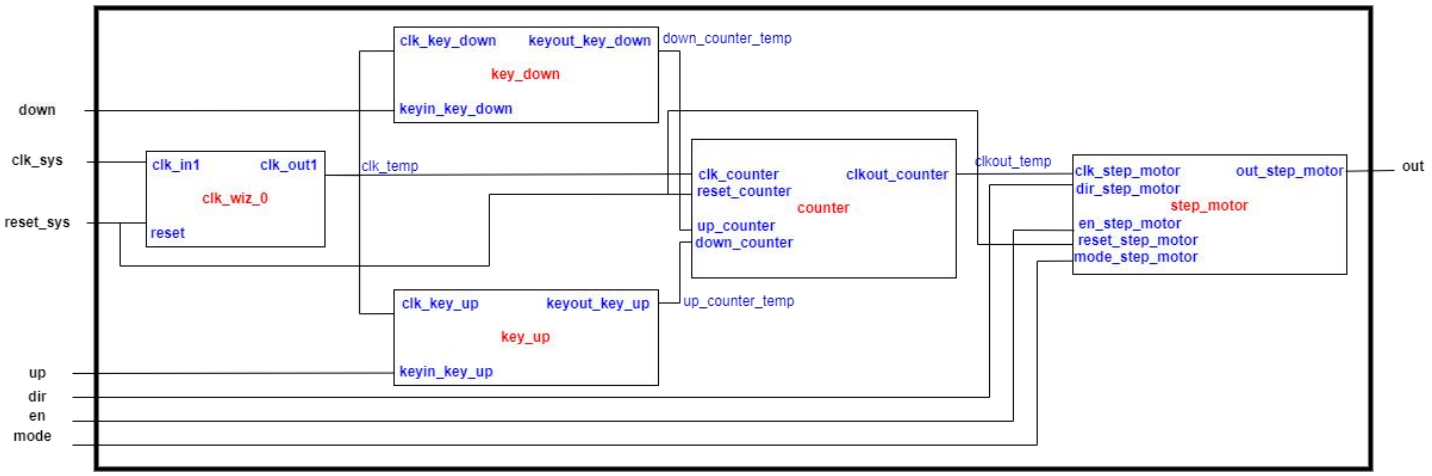
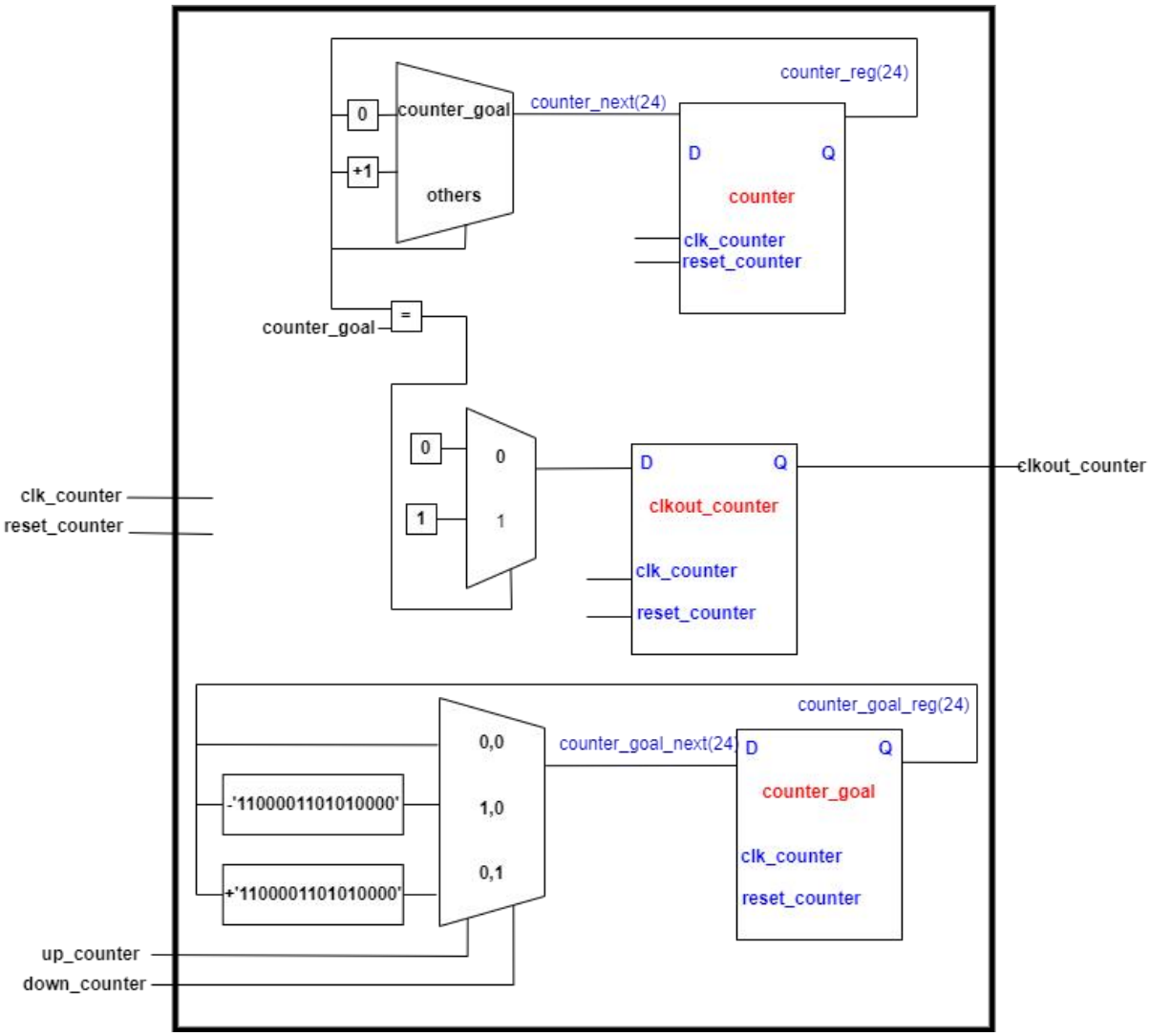
Step motor

11911214 杨鸿嘉

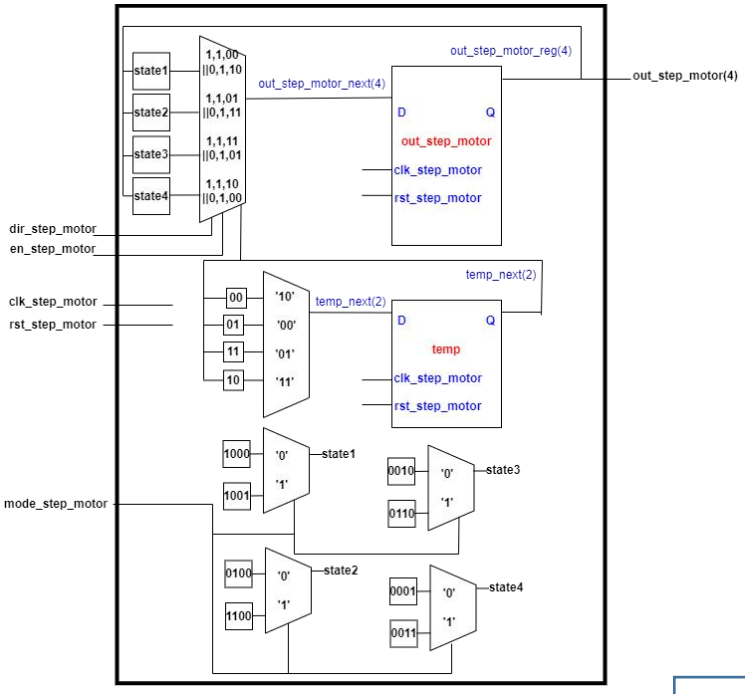
1.Top Design Structure Diagram and Block Diagram



Block Diagram for Counter:



Block Diagram for Step_motor



2. VHDL code for step_motor and Testbench code

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity counter is
Port(clk_counter,reset_counter: in std_logic;
       up_counter,down_counter:in std_logic;
      clkout_counter: out std_logic);
end counter;

architecture Behavioral of counter is
 signal counter: std_logic_vector(23 downto 0);
 signal counter_goal: std_logic_vector(23 downto 0);

begin
process (clk_counter,reset_counter) is
   begin
   if reset_counter = '1' then
      counter <= "000000000000000000000000";
    --counter_goal<="000011101000010010000000";
      counter_goal<="000000000000001111101000";
   elsif clk_counter'event and clk_counter='1' then
       if up_counter='1' then
            counter_goal<=counter_goal-"1100001101010000";
       elsif down_counter='1' then
            counter_goal<=counter_goal+"1100001101010000";
       else
       end if;

       if counter=counter_goal then
         counter<="000000000000000000000000";
         clkout_counter<='1';
       else
          counter<=counter+'1';
          clkout_counter<='0';
       end if;
   end if;
end process;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity key_down is
Port (clk_key_down,keyin_key_down:in std_logic;
          keyout_key_down:out std_logic );
end key_down;

architecture Behavioral of key_down is
signal count:integer;
begin
process(clk_key_down) is
begin
if(clk_key_down'event and clk_key_down='1') then
   if(keyin_key_down='1') then
          count<=count+1;
          if(count=10000) then
                    keyout_key_down<='1';
          else
                    keyout_key_down<='0';
          end if;
   else
          count<=0;
   end if;
end if;
end process;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity key_up is
Port (clk_key_up,keyin_key_up:in std_logic;
           keyout_key_up:out std_logic );
end key_up;

architecture Behavioral of key_up is
signal count:integer;
begin
process(clk_key_up) is
begin
if(clk_key_up'event and clk_key_up='1') then
   if(keyin_key_up='1') then
          count<=count+1;
          if(count=10000) then--对应现实 20ms
                    keyout_key_up<='1';
          else
                    keyout_key_up<='0';
          end if;
   else
          count<=0;
   end if;
end if;
end process;
end Behavioral;
```

Implementation for Elimination Buffeting of Keystroke:

Key_up and key_down is for Elimination Buffeting of Keystroke. The logic for this process is for each clk input, if the input of the button is in high level, use a counter to count the number of clk period with high level input. If this counter has value larger than 10000, this means that the button has been pushed and with only one time push, set the output signal high level. We through the elimination buffeting of keystroke problem through this method.

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity step_motor is
Port(clk_step_motor,dir_step_motor,en_step_motor,rst_step_motor:in std_logic;
      mode_step_motor: in std_logic;
      out_step_motor: out std_logic_vector(3 downto 0));--ABCD
end step_motor;

architecture Behavioral of step_motor is
signal temp:std_logic_vector(1 downto 0);
signal state1,state2,state3,state4:std_logic_vector(3 downto 0);
begin

process(clk_step_motor,dir_step_motor,en_step_motor,rst_step_motor) is
begin
if rst_step_motor= '1' then
      out_step_motor<="0000";
      temp<="00";
elsif clk_step_motor'event and clk_step_motor='1' then
   if en_step_motor='1' then
            if dir_step_motor='1' then
               if(temp="00") then
                  out_step_motor<=state1;
                  temp<="01";
               elsif temp="01" then
                  out_step_motor<=state2;
                  temp<="11";
               elsif temp="11" then
                  out_step_motor<=state3;
                  temp<="10";
               else
                  out_step_motor<=state4;
                  temp<="00";
               end if;
            else
            if(temp="00") then
              out_step_motor<=state4;
              temp<="01";
            elsif temp="01" then
              out_step_motor<=state3;
              temp<="11";
            elsif temp="11" then
              out_step_motor<=state2;
              temp<="10";
            else
               out_step_motor<=state1;
               temp<="00";
            end if;
      end if;
   else
      out_step_motor<="0000";
      temp<="00";
   end if;
end if;
end process;

state1<="1000" when mode_step_motor='0' else "1001";
state2<="0100" when mode_step_motor='0' else "1100";
state3<="0010" when mode_step_motor='0' else "0110";
state4<="0001" when mode_step_motor='0' else "0011";
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity step_motor_top is
Port (clk_sys,rst_sys,en_sys,dir_sys,down_sys,up_sys,mode_sys:in std_logic;--'0' wave   '1' fullstep
        out_sys:out std_logic_vector(3 downto 0));
end step_motor_top;

architecture structure of step_motor_top is
component clk_wiz_0
  port(clk_out1: out std_logic;reset: in std_logic;clk_in1: in std_logic);
end component;
component counter
Port(clk_counter,reset_counter: in std_logic;
    up_counter,down_counter:in std_logic;
     clkout_counter: out std_logic);
end component;
component step_motor is
Port(clk_step_motor,dir_step_motor,en_step_motor,rst_step_motor:in std_logic;
       mode_step_motor: in std_logic;
       out_step_motor: out std_logic_vector(3 downto 0));--ABCD
end component;
component key_down is
Port (clk_key_down,keyin_key_down:in std_logic;
         keyout_key_down:out std_logic );
end component;
component key_up is
Port (clk_key_up,keyin_key_up:in std_logic;
         keyout_key_up:out std_logic );
end component;

signal clk_temp,clkout_temp,up_counter_temp,down_counter_temp:std_logic;


begin
clkwiz : clk_wiz_0 port map (clk_in1 => clk_sys,reset => rst_sys,clk_out1=> clk_temp);
stepmotor:step_motor port map
(clk_step_motor=>clkout_temp,dir_step_motor=>dir_sys,en_step_motor=>en_sys,rst_step_motor=>rst_
sys,out_step_motor=>out_sys, mode_step_motor=>mode_sys);
counter_step: counter port map
(clk_counter=>clk_temp,reset_counter=>rst_sys,clkout_counter=>clkout_temp,up_counter=>up_counte
r_temp,down_counter=>down_counter_temp);
keyup:key_up port map
(clk_key_up=>clk_temp,keyin_key_up=>up_sys,keyout_key_up=>up_counter_temp);
keydown:key_down port map
(clk_key_down=>clk_temp,keyin_key_down=>down_sys,keyout_key_down=>down_counter_temp);

end structure;
```

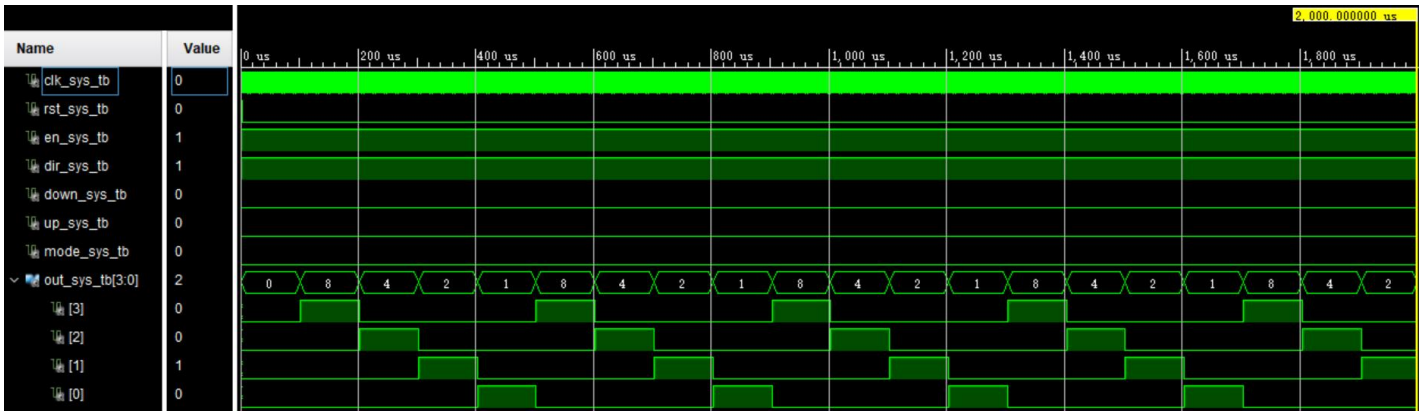
Testbench code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity key_down_tb is
end key_down_tb;

architecture behavior of key_down_tb is
component key_down
Port (clk_key_down,keyin_key_down:in std_logic;
          keyout_key_down:out std_logic );
end component;

signal clk_key_down_tb,keyin_key_down_tb,keyout_key_down_tb: std_logic;

begin
    uut: key_down PORT MAP (clk_key_down=>clk_key_down_tb,keyin_key_down=>keyin_key_down_tb,keyout_key_down=>keyout_key_down_tb);

    keyin_key_down_tb<='0', '1' after 1ms,'0' after 1.005 ms,'1' after 1.010ms, '0' after 1.015ms, '1' after 1.020ms,'0' after 1.025ms,'1' after 1.03ms,'0' after 1.035ms, '1' after
1.040ms, '0'after 1.045ms, '1' after 1.05ms;

    clock_gen: process
    constant period : time := 100 ns;
    begin
       clk_key_down_tb <= '0';
       wait for period/2;
       clk_key_down_tb <= '1';
       wait for period/2;
    end process;

end behavior;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity step_motor_top_tb is
end step_motor_top_tb;

architecture behavior of step_motor_top_tb is
component step_motor_top
Port (clk_sys,rst_sys,en_sys,dir_sys,down_sys,up_sys,mode_sys:in std_logic;--'0' wave    '1' fullstep
          out_sys:out std_logic_vector(3 downto 0));
end component;

signal clk_sys_tb,rst_sys_tb,en_sys_tb,dir_sys_tb,down_sys_tb,up_sys_tb,mode_sys_tb: std_logic;
signal out_sys_tb:std_logic_vector(3 downto 0);

begin
    uut: step_motor_top PORT MAP (clk_sys=>clk_sys_tb,rst_sys=>rst_sys_tb,en_sys=>en_sys_tb,dir_sys=>dir_sys_tb,out_sys=>out_sys_tb,
                                  down_sys=>down_sys_tb,up_sys=>up_sys_tb,mode_sys=>mode_sys_tb );

    rst_sys_tb<='0', '1' after 10 ns,'0' after 50 ns;
    dir_sys_tb<='1';
    en_sys_tb<='1';
    down_sys_tb<='0';
    up_sys_tb<='0';
    mode_sys_tb<='0';

    clock_gen: process
    constant period : time := 10 ns;
    begin
       clk_sys_tb <= '0';
       wait for period/2;
       clk_sys_tb <= '1';
       wait for period/2;
    end process;

end behavior;
```

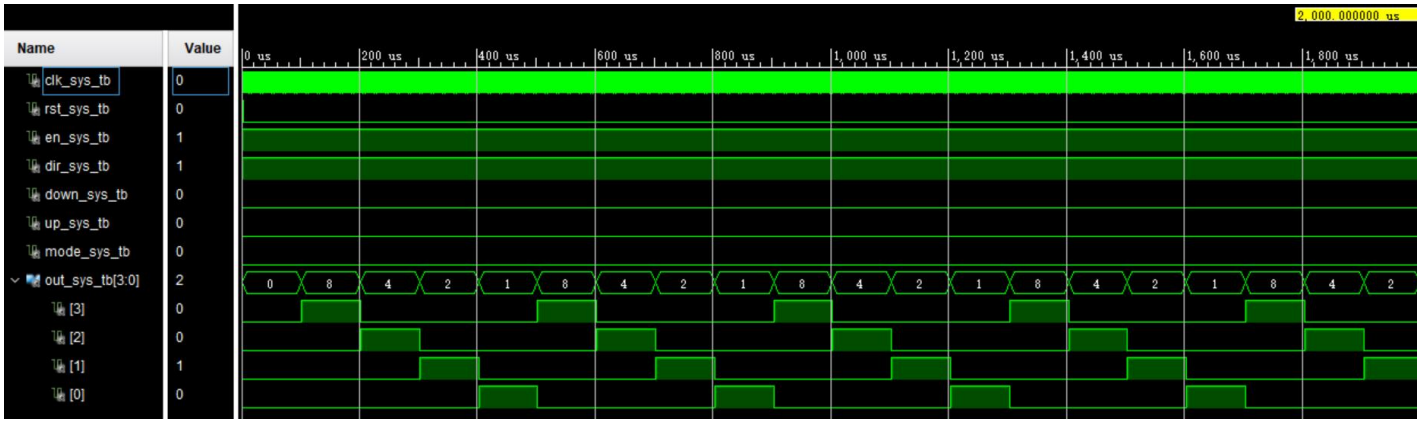
3. Simulation Result

Behavior simulation

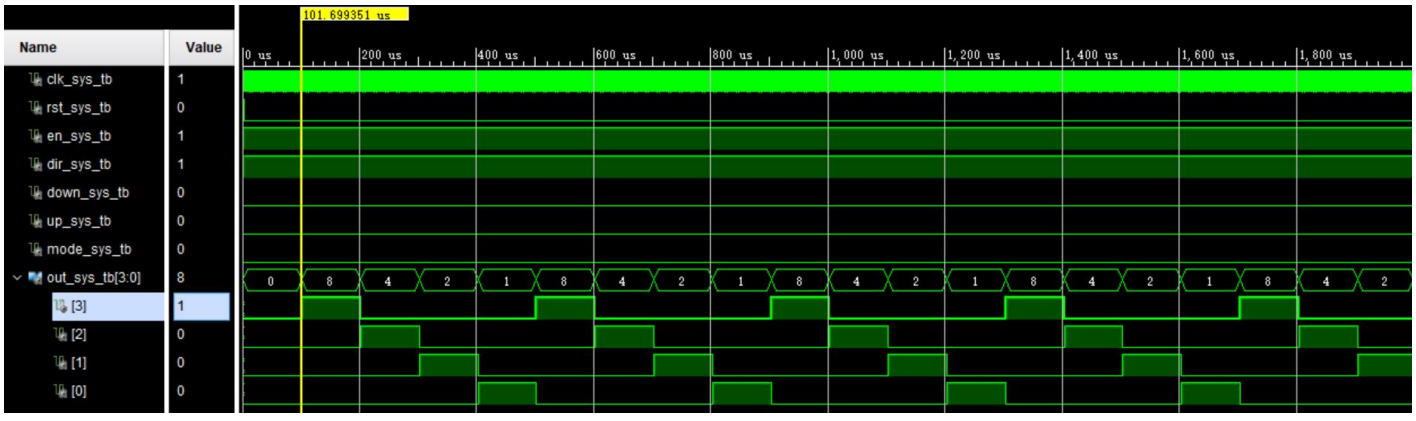


After reset, the program work well according to our setting, we can see the output waveform of the step motor.

Post-synthesis functional simulation:

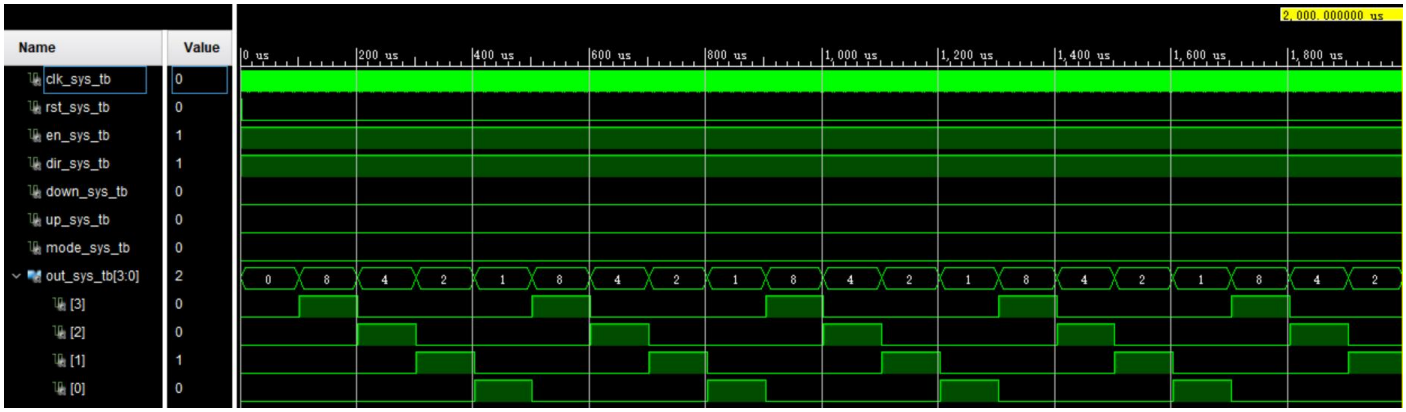


Post-synthesis timing simulation:

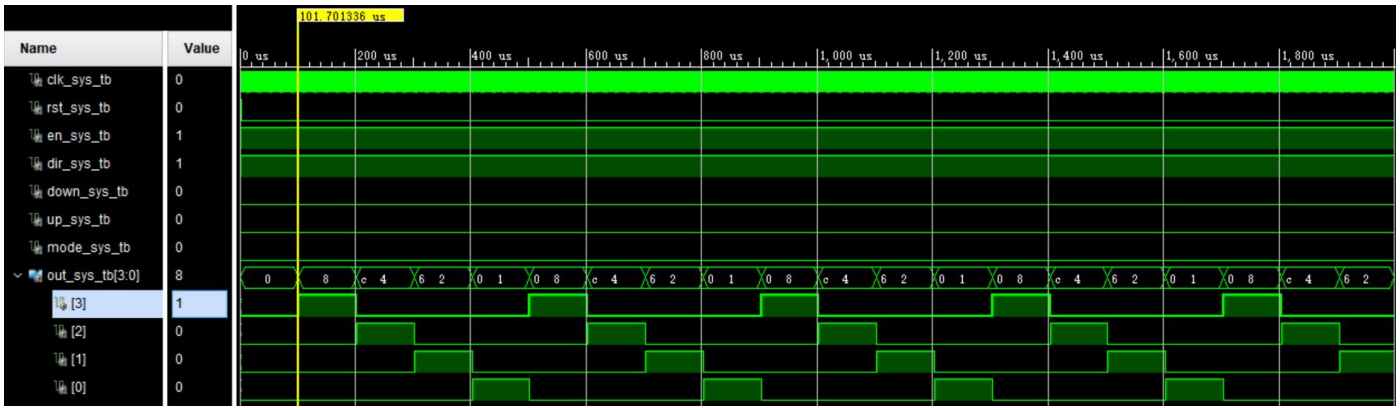


For the post-synthesis timing simulation result, we can find the delay for the output is nearly 1.699us.

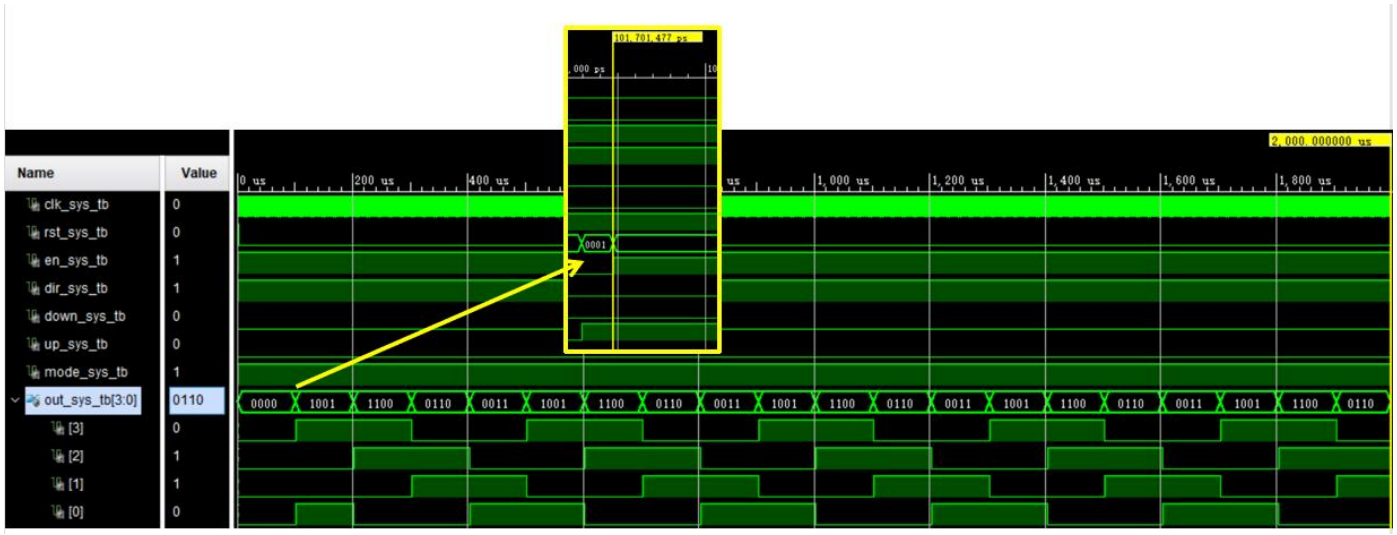
Post-Implementation functional simulation:



Post-Implementation timing simulation:

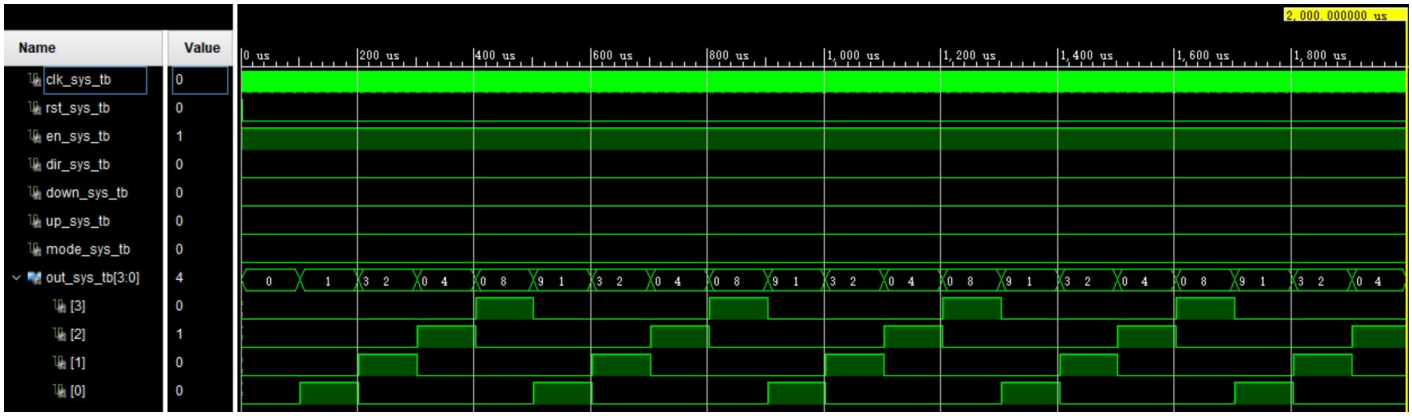


From the post-implementation timing simulation, the delay for the output is 1.70us, which is larger than post-synthesis simulation. We can see the competitive adventure phenomenon and the output waveform for step motor has delay for near step, that is:

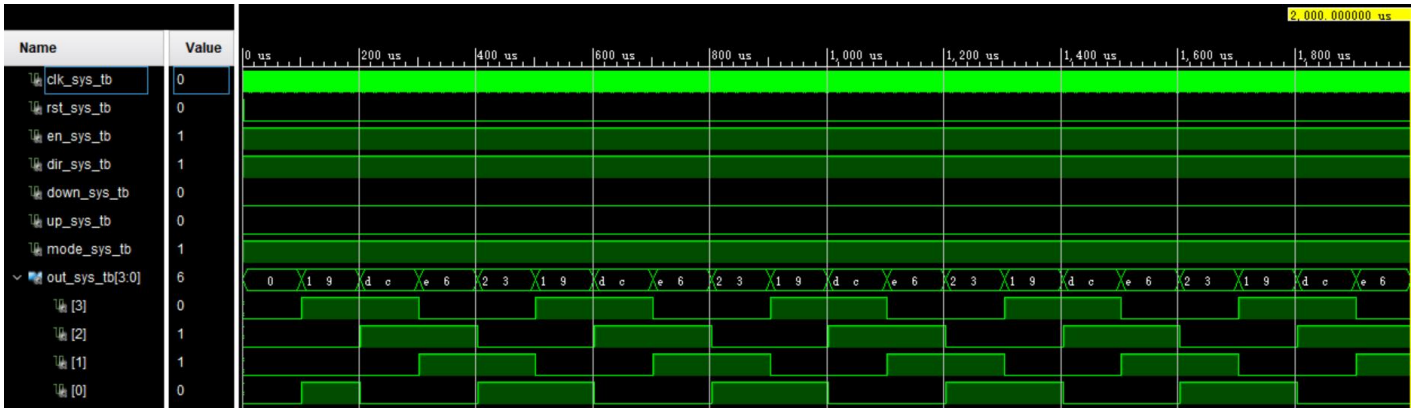


We can see that the four step are not closely alternating, there may be slight overlap and neutral period. But this will not influence our actual performance.

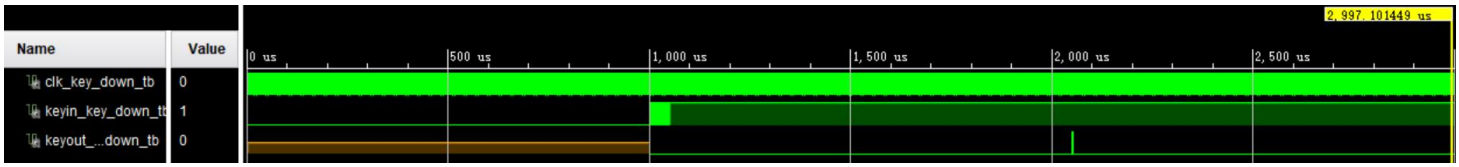
Different direction post-implementation timing simulation:



Full step drive post-implementation timing simulation:

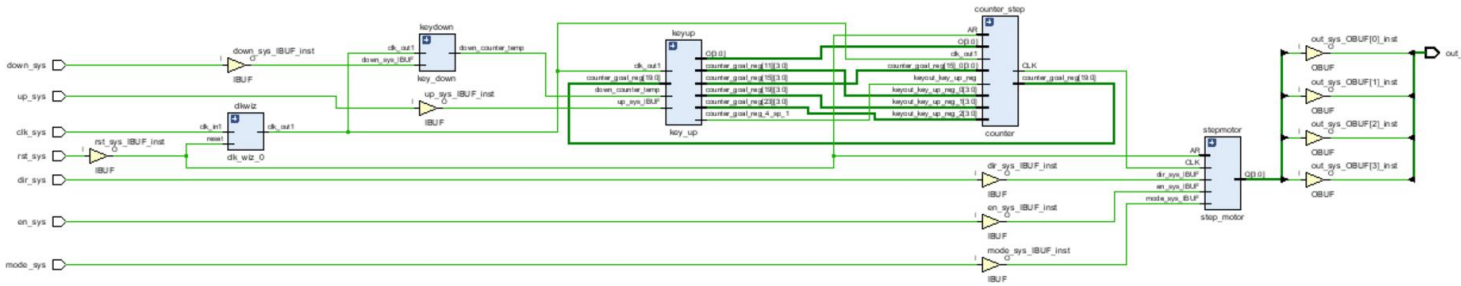


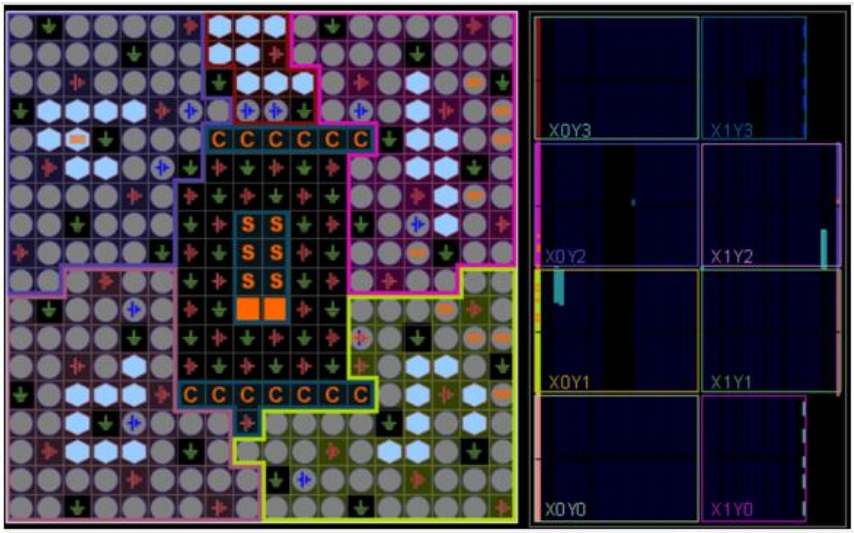
Testing for the Elimination Buffeting of Keystroke:



We can find that the Elimination Buffeting of Keystroke work as our expect.

4 Simulation Resource:





Utilization Post-Synthesis | **Post-Implementation** Graph | **Table**

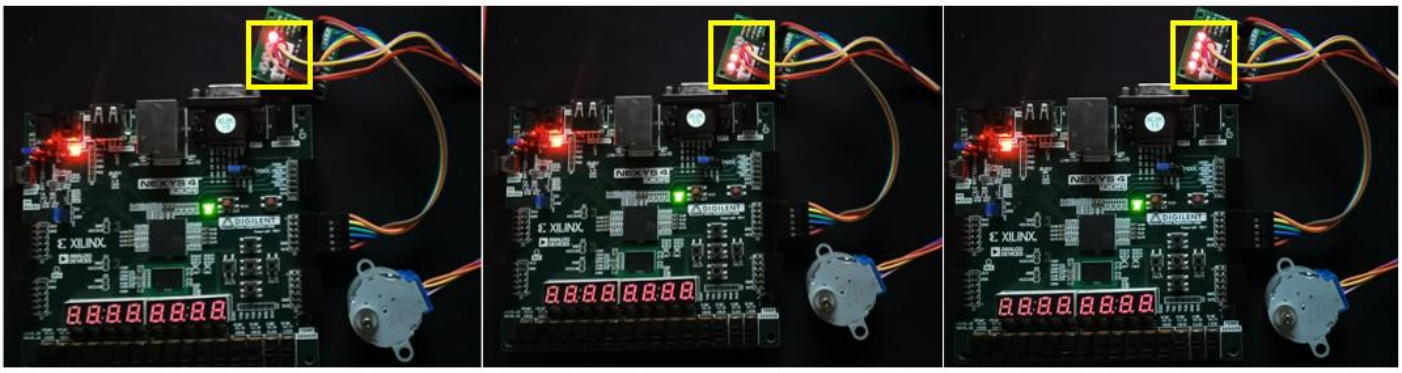
Resource	Utilization	Available	Utilization %
LUT	74	63400	0.12
FF	117	126800	0.09
IO	11	210	5.24
BUFG	2	32	6.25
MMCM	1	6	16.67

Power **Summary** | On-Chip

Total On-Chip Power:	**0.191 W**
Junction Temperature:	**25.9 °C**
Thermal Margin:	59.1 °C (12.8 W)
Effective ϑJA:	4.6 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

5 Actual Testing



Wave drive **Full step drive** **Full step drive(faster)**