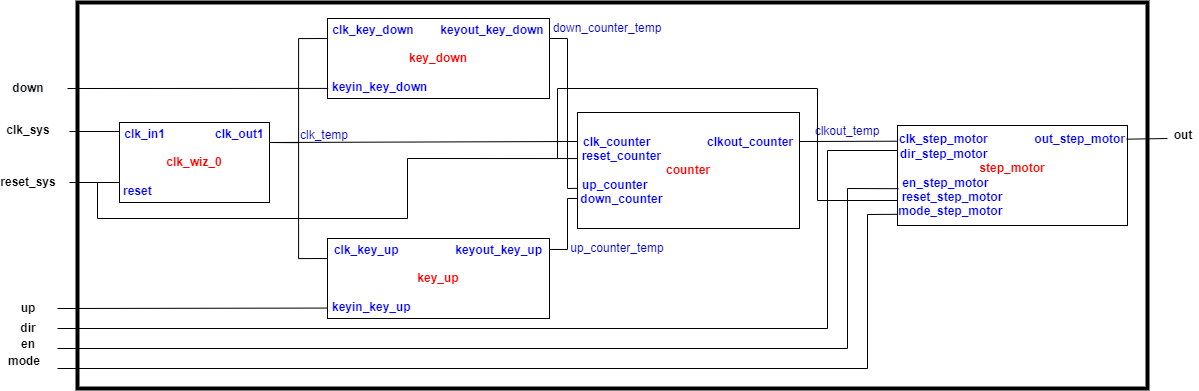
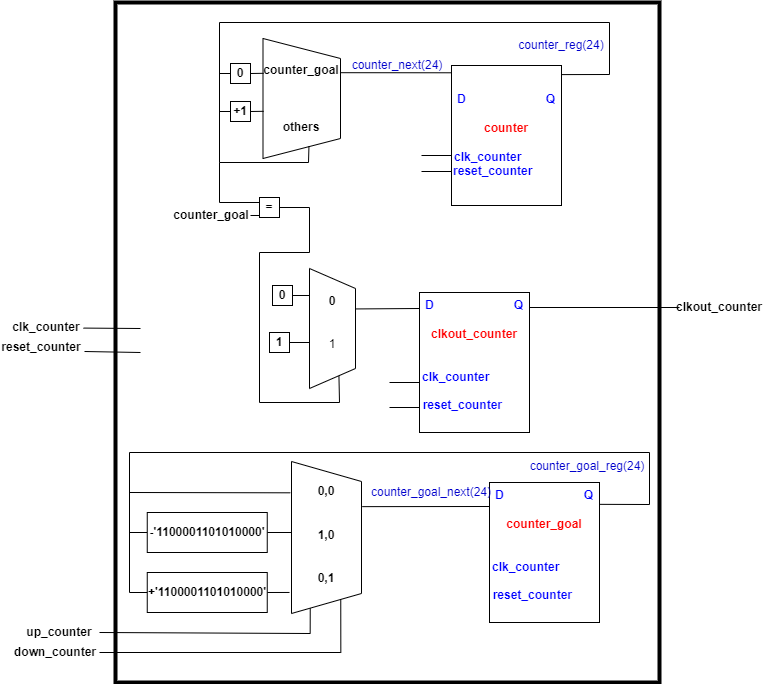
**Step motor**

11911214 杨鸿嘉

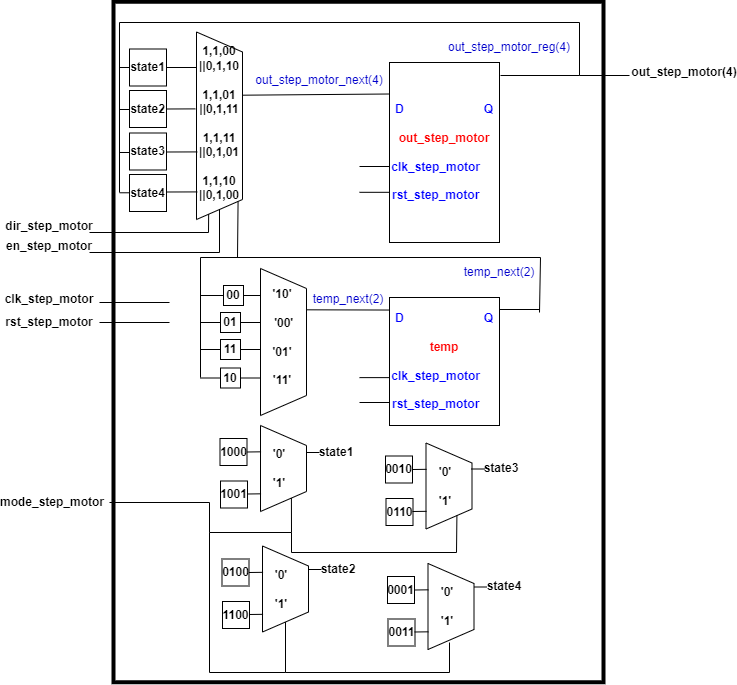
**1.Top Design Structure Diagram and Block Diagram**



Block Diagram for Counter:



Block Diagram for Step\_motor



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;

1. **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<="000011110100001001000000";**

**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\_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;

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\_counter\_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;

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;

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';

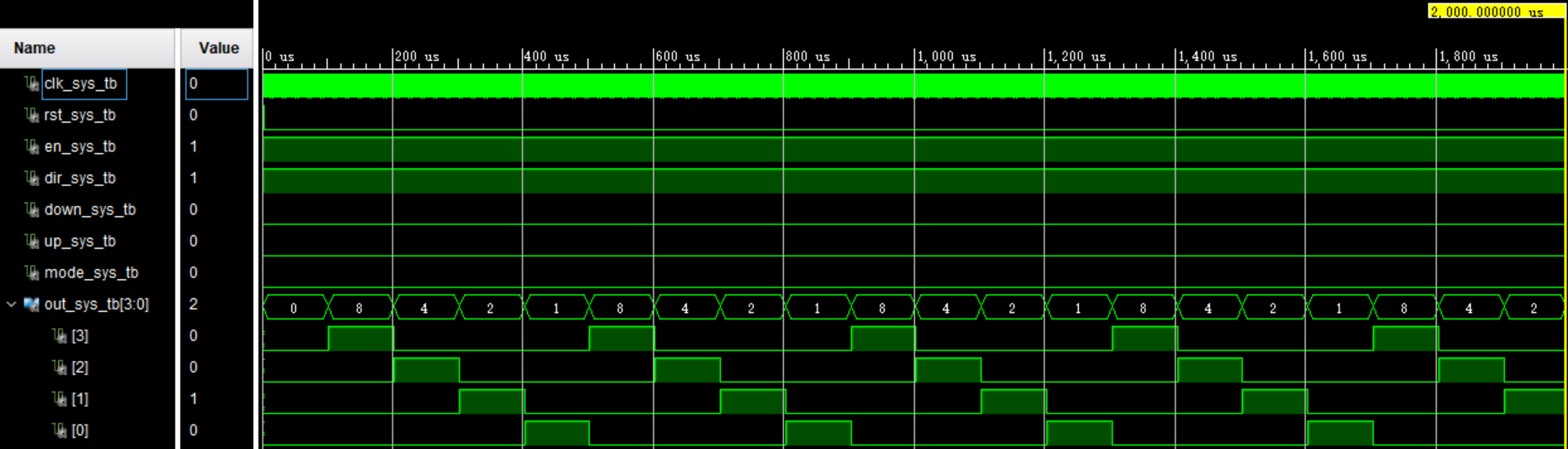
wait for period/2;

end process;

end behavior;

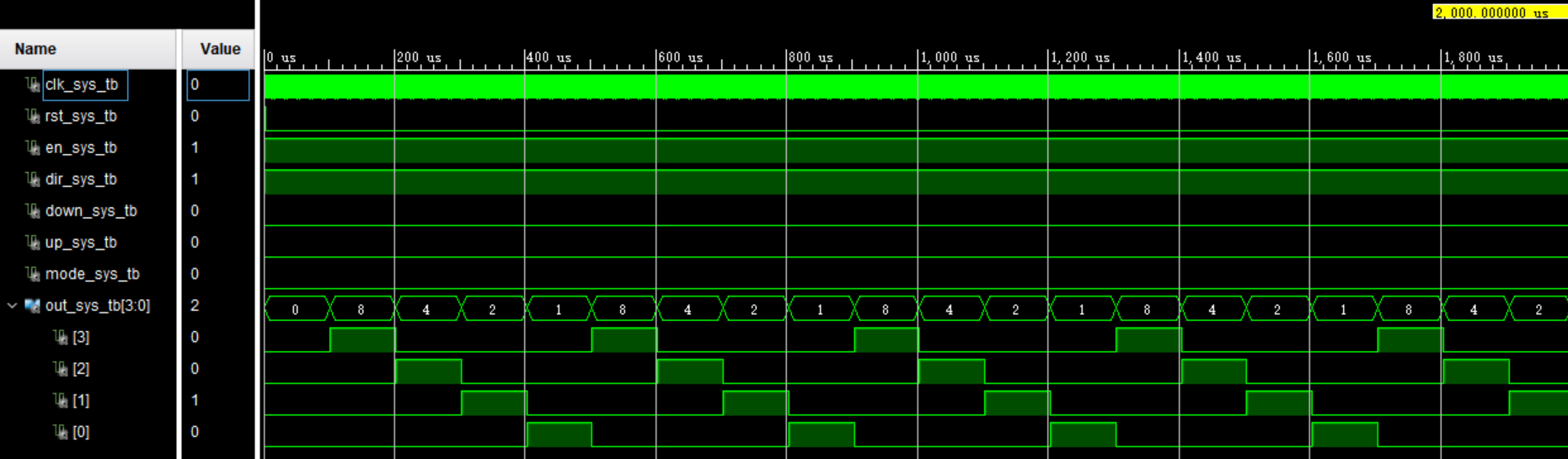
1. **Simulation Result**

**Behavior simulation**

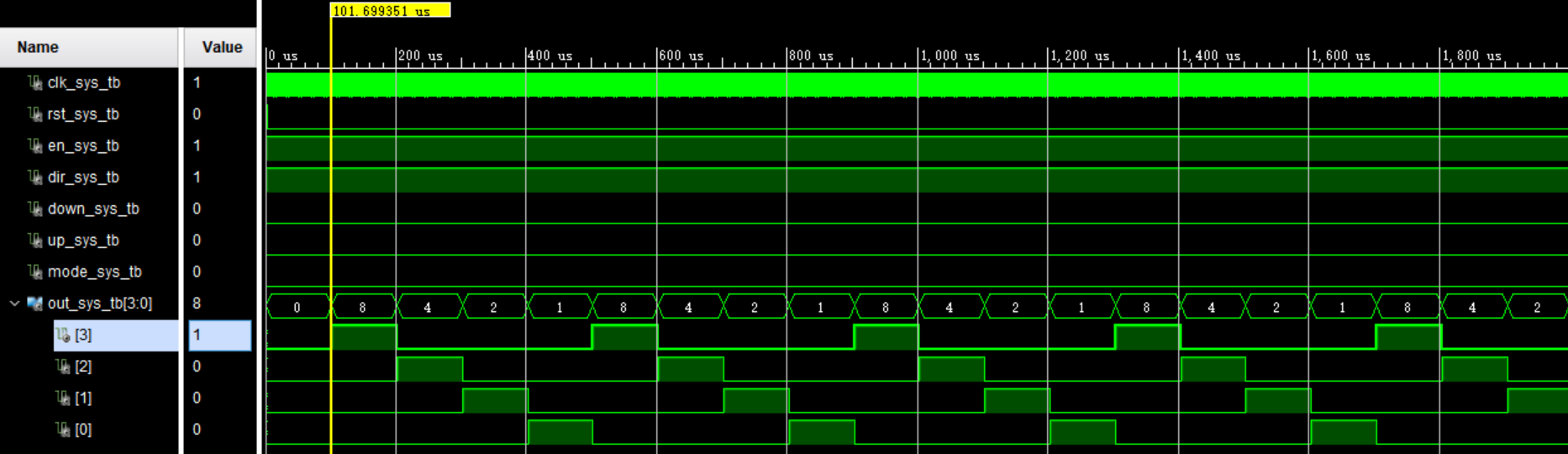


After reset, the program work well according to our setting, we can see the output waveform of the step motor.

**Post-synthesis functional simulation:**

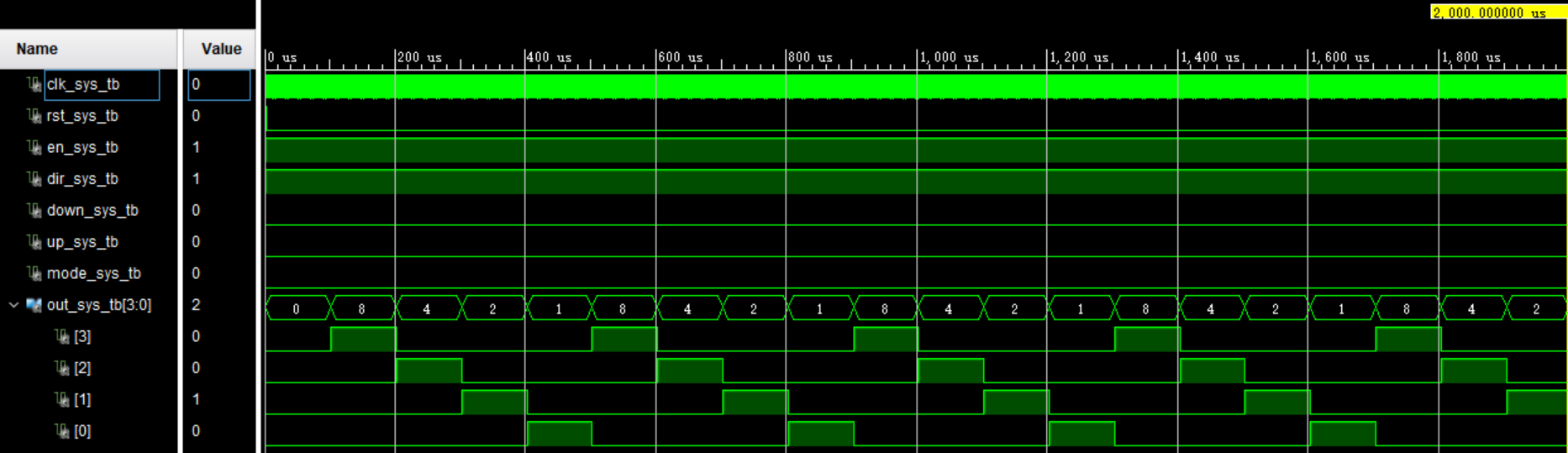


**Post-synthesis timing simulation:**

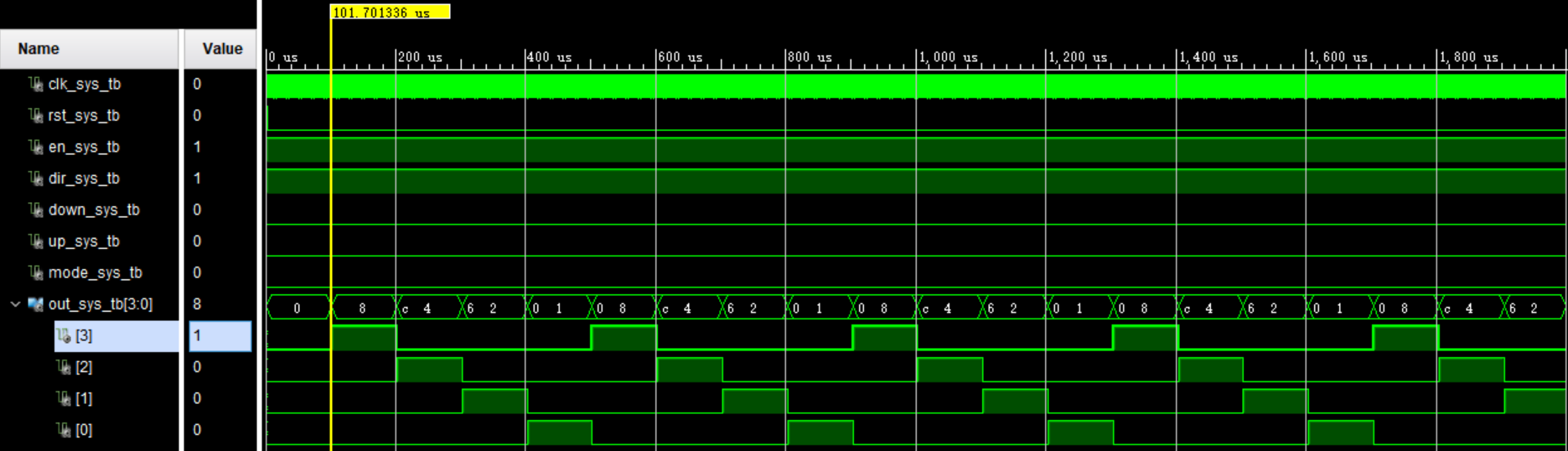


For the post-synthesis timing simulation result, we can find the delay for the output is nearly 1.699us.

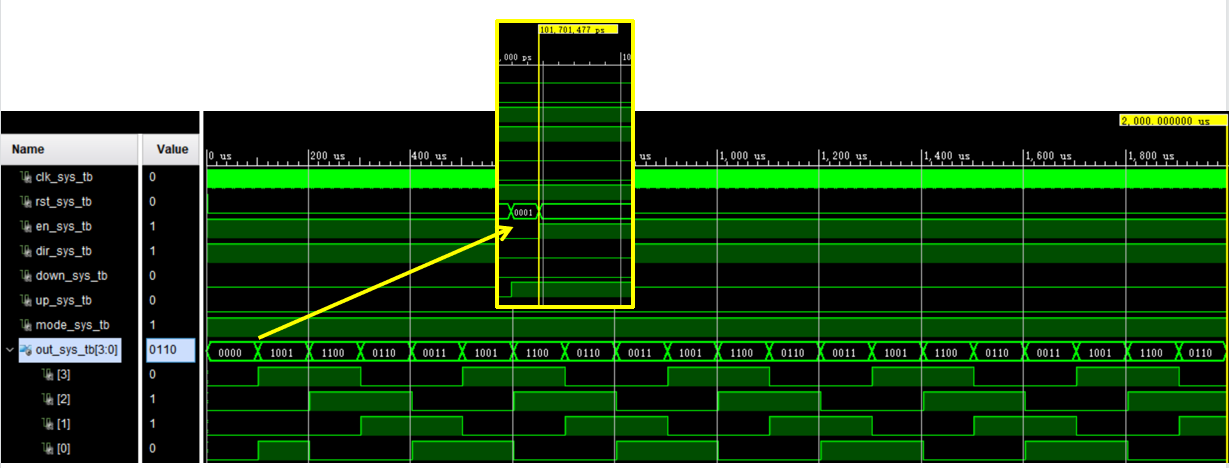
**Post-Implementation functional simulation:**



**Post-Implementation timing simulation:**

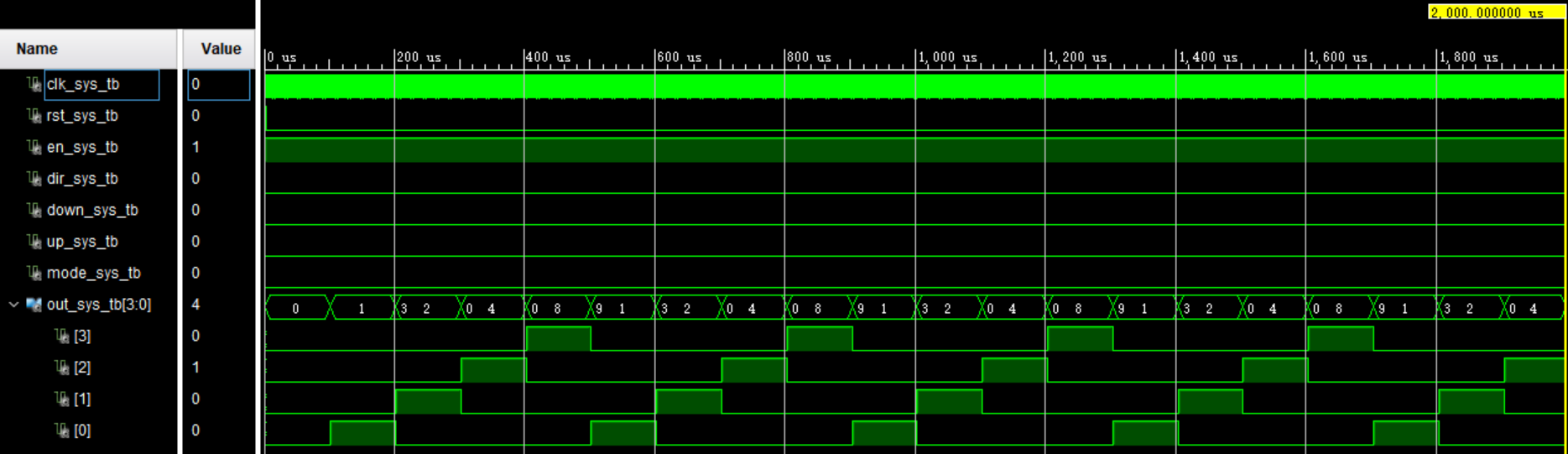


From the post-implementation timing simulation, the delay for the output is 1.70us, which is larger than post-synthesis simulation. We can see the competitive adventure phenomenon and the output waveform for step motor has delay for near step, that is:

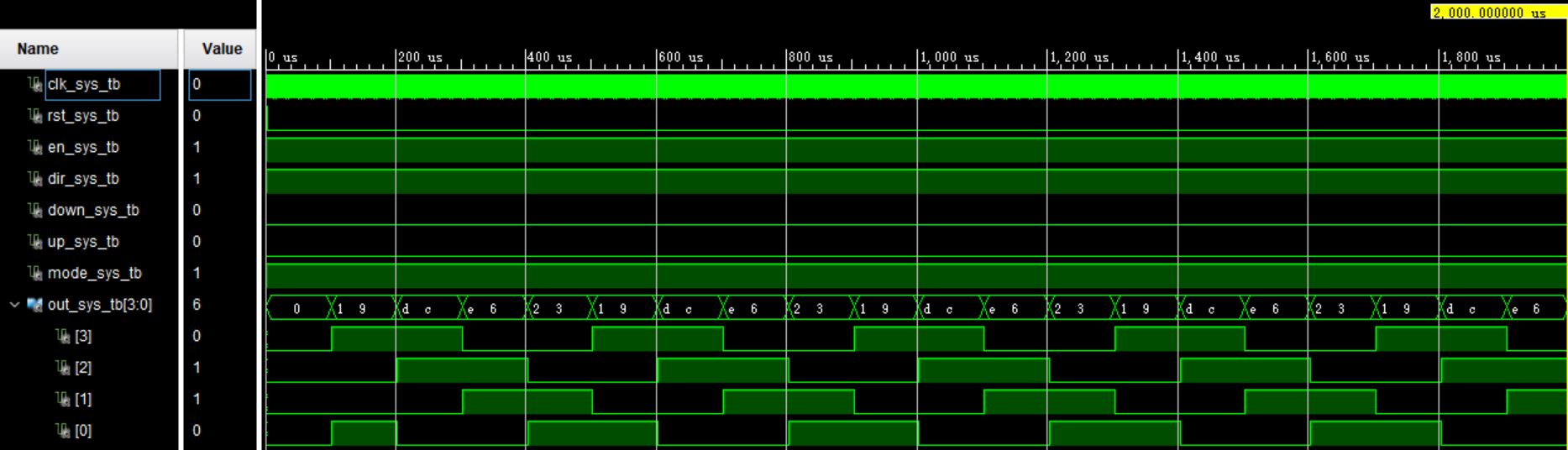


We can see that the four step are not closely alternating, there may be slight overlap and neutral period. But this will not influence our actual performance.

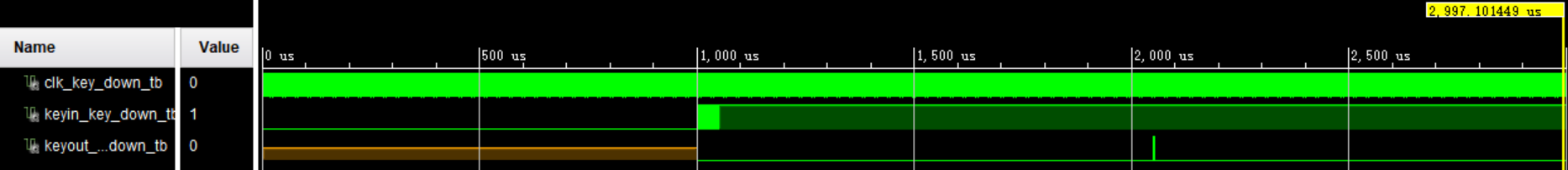
**Different direction post-implementation timing simulation:**



**Full step drive post-implementation timing simulation:**

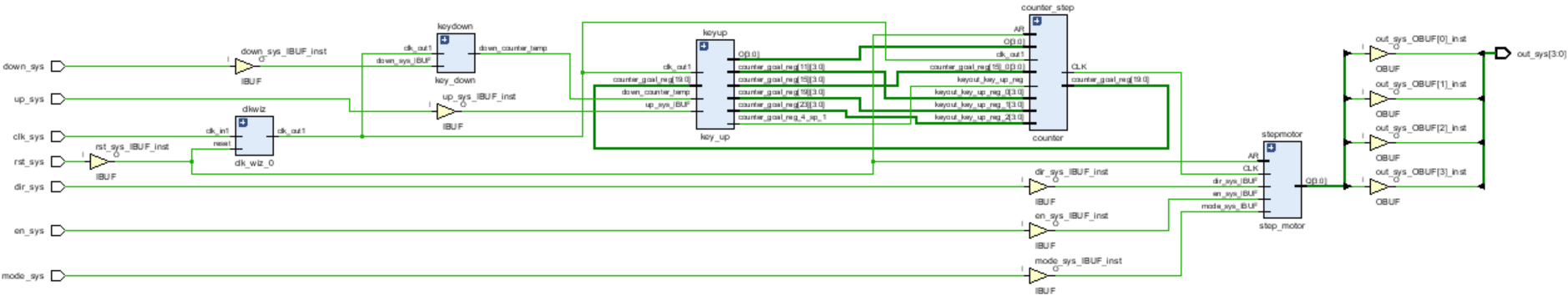


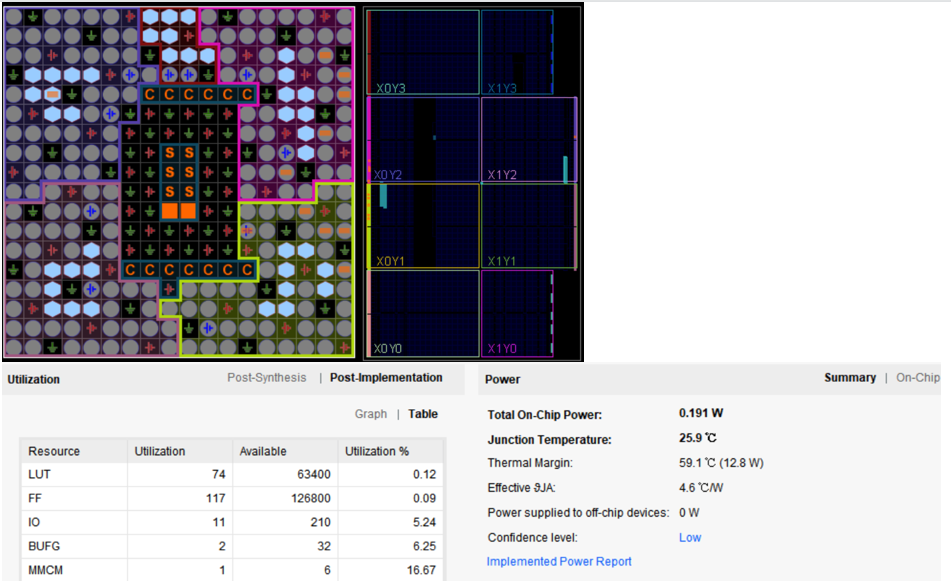
**Testing for the Elimination Buffeting of Keystroke:**



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

**4 Simulation Resource:**





**5 Actual Testing**

