Exercise4 Design and realize a three-digit decimal counter

张旭东 12011923

1. Introduction

The purpose of this lab is to design and realize a three-digit decimal counter. The design has some important functions, including an asynchronous reset function, a synchronous load function to set a start number of the counting, the function of increasing 1 for every 1 second and function of displaying the counting results on the 12 LEDs. The asynchronous reset function is realized by using a push button to initialize the counter to 000. The synchronous load function is realized by using a switch to turn on or turn off the load mode and use 12 switches to set the start number.

2.Lab result and analysis

2.1 Prelab exercise:

The FPGA board provides only a 100MHz clock source, which is 10^8 times faster than the required clock. So I have designed a component of a frequency divider that slow down the output by 10^8 times for a clock input. The conceptual diagram of the frequency divider is in `Fig 1`.

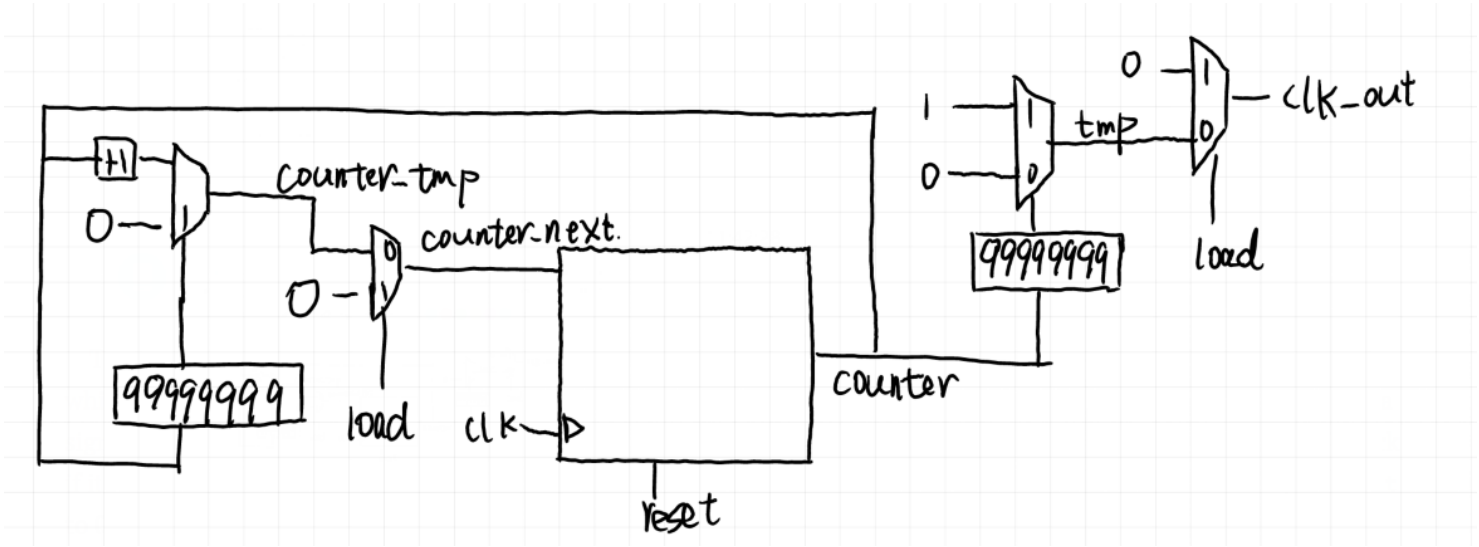


Fig.1 conceptual diagram of the frequency divider

The component of frequency divider has three input ports, which are `clk`,`load` and `reset` and one output port, which is `clk_out`. Because the clock source of FPGA board is 100MHz and what we need is the clock of 1Hz, we define a signal `counter` and it increases by 1 when the rising edge of clock reaches. The purpose of input ports, `load` and `reset` is to satisfy satisfies asynchronous reset function and synchronous load function of the whole three-digit decimal counter. For flip-flop, when the logic value of `reset` is 1, set `counter` to 00000000000000000000000000. When the rising edge of clock reaches and the logic value of `reset` is 0, the value of `counter_next` is assigned to `counter`. For synchronous, check if `counter` equals 99999999. If yes, set `counter_tmp` to 0 and set `tmp` to 1. If no, set `counter_tmp` to `counter+1` and set `tmp` to 0. When the logic value of `load` is 1, set `counter_next` and `clk_out` to 0. Otherwise, `counter_tmp` is assigned to `counter_next` and `tmp` is assigned to `clk_out`. This can generate the clock of 1Hz. (The detail of this can be shown in the following code).

The VHDL code of the frequency divider is as below:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.all;
entity frequency_divider is
    Port ( CLK : in STD_LOGIC;
           reset: in STD_logic;
           load: in STD_LOGIC;
          clk_out : out STD_LOGIC);
```

```
end frequency_divider;

architecture Behavioral of frequency_divider is
    signal counter:STD_LOGIC_VECTOR(26 DOWNTO 0);
    signal counter_next:STD_LOGIC_VECTOR(26 DOWNTO 0);
    signal counter_tmp:STD_LOGIC_VECTOR(26 DOWNTO 0);
    signal tmp:STD_LOGIC;
begin
    process(CLK,reset)
    begin
    if reset='1'then
        counter<="00000000000000000000000000";
    elsif CLK'event and CLK='1' then
        counter<=counter_next;
    end if;
    end process;

    counter_tmp<="00000000000000000000000000" when counter=99999999 else
                 counter+1;

    counter_next<="00000000000000000000000000" when load='1' else
                  counter_tmp;
    tmp<='1' when counter=99999999 else
         '0';
    clk_out<='0' when load='1' else
             tmp;
end Behavioral;
```

2.2 Lab exercise

There are two ways to make use of the output of the frequency divider in my design. One is to use its output signal as the clock and the other is to use it as an enable signal. The first one actually is an asynchronous design because the registers in the circuits are controlled by more than 1 clock, which should be avoided especially in a high-speed circuit design. So, the output of the frequency divider should be used as an enable signal. That means the component of 3-digit decimal counter should have an input port of `enable`. The conceptual diagram of the 3-digit decimal counter is in `Fig 2`.

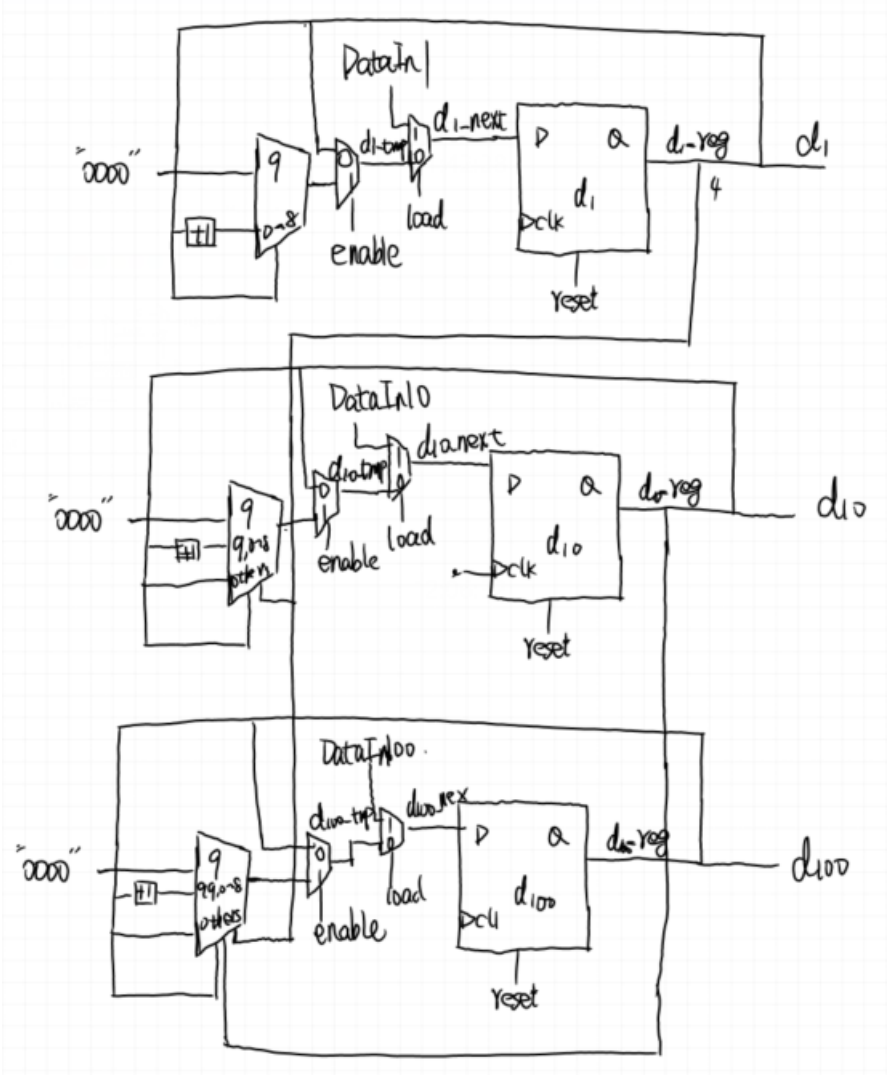


Fig.2 conceptual diagram of component of 3-digit decimal counter

The component of 3-digit decimal counter has seven input ports, which are `clk`,`load`, `reset`, `enable`,`DataIn1` ,`DataIn10` and `DataIn100` and three output ports, which are `d1`, `d10` and `d100`. The purpose of input ports, `load` and `reset` is to satisfy satisfies asynchronous reset function and synchronous load function of the whole three-digit decimal counter. For flip-flop, the input ports of `reset` set `d1`, `d10` and `d100` to 0000, 0000 and 0000 when the value of `reset` is 1, which satisfies asynchronous reset function. When the rising edge of clock reaches and the logic value of `reset` is 0, the value of `d1_next`, `d10_next` and `d100_next` is assigned to `d1_reg`, `d10_reg` and `d100_reg`. For synchronous, when the logic value of `enable` is 1, `d1_reg`, `d10_reg` and `d100_reg` is assigned to `d1_tmp`, `d10_tmp` and `d100_tmp`. Otherwise, `d1_tmp`, `d10_tmp` and `d100_tmp` are assigned the corresponding value according to the corresponding judgment condition. When the logic value of `load` is 1, `DataIn1`, `DataIn10` and `DataIn100` is assigned to `d1_next`, `d10_next` and `d100_next`. Otherwise, `d1_tmp`, `d10_tmp` and `d100_tmp` is assigned to `d1_next`, `d10_next` and `d100_next`. What's

more, if the value of `DataIn1` loaded is greater than 9, `d1_next` is assigned to 0000 and `d10_next` is assigned to `DataIn10+1`. This is similar for `DataIn10`. For `DataIn100`, if the value of `DataIn100` loaded is greater than 9, `d100_next` is assigned to 0000. (The detail of this can be shown in the following code).

The VHDL code of the component of 3-digit decimal counter is as below:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.all;

entity counter_new is
    Port ( CLK : in STD_LOGIC;
           reset : in STD_LOGIC;
           load : in STD_LOGIC;
           enable: in STD_LOGIC;
           DataIn1 : in STD_LOGIC_VECTOR (3 downto 0);
           DataIn10 : in STD_LOGIC_VECTOR (3 downto 0);
           DataIn100 : in STD_LOGIC_VECTOR (3 downto 0);
           d1 : out STD_LOGIC_VECTOR (3 downto 0);
           d10 : out STD_LOGIC_VECTOR (3 downto 0);
           d100 : out STD_LOGIC_VECTOR (3 downto 0));
end counter_new;

architecture concurrent_arch of counter_new is
    signal d1_reg,d10_reg,d100_reg:std_logic_vector(3 downto 0);
    signal d1_next,d10_next,d100_next:std_logic_vector(3 downto 0);
    signal d1_tmp,d10_tmp,d100_tmp:std_logic_vector(3 downto 0);

begin
    process(CLK,reset)is
    begin
    if reset='1'then
        d1_reg<="0000";
        d10_reg<="0000";
```

```
        d100_reg<="0000";
    elsif CLK'event and CLK='1'then
            d1_reg<=d1_next;
            d10_reg<=d10_next;
            d100_reg<=d100_next;
    end if;
end process;

    d1_tmp<=d1_reg when enable='0' else
        "0000" when d1_reg=9  else
        d1_reg+1;
    d10_tmp<=d10_reg when enable='0' else
        "0000" when (d1_reg=9 and d10_reg=9 ) else
        d10_reg+1 when d1_reg=9 else
        d10_reg;
    d100_tmp<=d100_reg when enable='0' else
        "0000" when (d1_reg=9 and d10_reg=9 and d100_reg=9) else
        d100_reg+1 when (d1_reg=9 and d10_reg=9 ) else
        d100_reg;

    d1_next<=d1_tmp when load='0' else
             "0000" when DataIn1>9 else
             DataIn1;
    d10_next<=d10_tmp when load='0' else
              "0000" when DataIn10>9 else
              DataIn10+1 when  DataIn1>9 else
              DataIn10;
    d100_next<=d100_tmp when load='0' else
               "0000" when  DataIn100>9 else
               DataIn100+1 when  DataIn10>9 else
               DataIn100;

    d1<=d1_reg;
    d10<=d10_reg;
    d100<=d100_reg;

end concurrent_arch;
```

The top level structural of the overall design is in `Fig 3`.

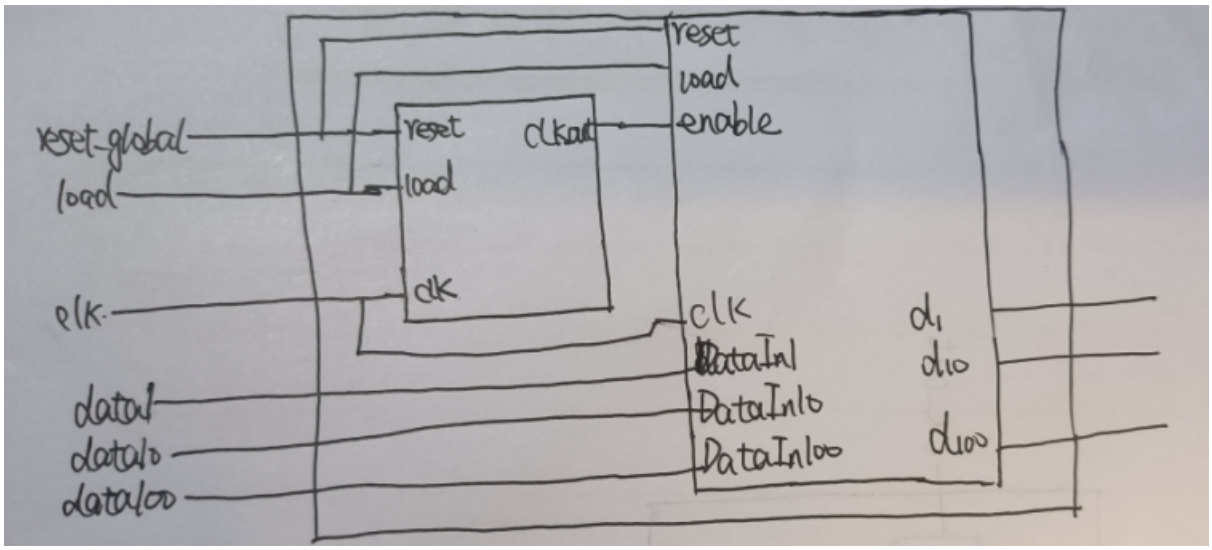


Fig.3 top level structural of the overall design

The purpose of input ports, `load` and `reset` is to satisfy satisfies asynchronous reset function and synchronous load function of the whole three-digit decimal counter.

The code of the top module is as below:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.all

entity clk_demical_counter is
    Port ( reset_global : in STD_LOGIC;
           clk : in STD_LOGIC;
           load : in STD_LOGIC;
           data10 : in STD_LOGIC_VECTOR (3 downto 0);
           data1 : in STD_LOGIC_VECTOR (3 downto 0);
           data100 : in STD_LOGIC_VECTOR (3 downto 0);
           d1 : out STD_LOGIC_VECTOR (3 downto 0);
           d10 : out STD_LOGIC_VECTOR (3 downto 0);
```

```
            d100 : out STD_LOGIC_VECTOR (3 downto 0));
end clk_demical_counter;

architecture Behavioral of clk_demical_counter is
    component counter_new is
        Port ( CLK : in STD_LOGIC;
           reset : in STD_LOGIC;
           load : in STD_LOGIC;
           enable: in STD_LOGIC;
           DataIn1 : in STD_LOGIC_VECTOR (3 downto 0);
           DataIn10 : in STD_LOGIC_VECTOR (3 downto 0);
           DataIn100 : in STD_LOGIC_VECTOR (3 downto 0);
           d1 : out STD_LOGIC_VECTOR (3 downto 0);
           d10 : out STD_LOGIC_VECTOR (3 downto 0);
           d100 : out STD_LOGIC_VECTOR (3 downto 0));
    end component;
    component frequency_divider is
         Port ( CLK : in STD_LOGIC;
            reset: in STD_logic;
            load: in STD_logic;
           clk_out : out STD_LOGIC);
    end component frequency_divider;
    signal s1:STD_LOGIC;

begin
    HA1:frequency_divider port
map(reset=>reset_global,CLK=>clk,clk_out=>s1,load=>load);
    HA2:counter_new port
map(CLK=>clk,reset=>reset_global,load=>load,DataIn1=>data1,DataIn10=>data10,Da
taIn100=>data100,d1=>d1,d10=>d10,d100=>d100,enable=>s1);



end Behavioral;
```

2.3 Test bench

The code of test bench is as below:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.all;

entity tb_clk_demical_counter is
--  Port ( );
end tb_clk_demical_counter;

architecture Behavioral of tb_clk_demical_counter is
    component clk_demical_counter is
        Port ( reset_global : in STD_LOGIC;
           clk : in STD_LOGIC;
           load : in STD_LOGIC;
           data10 : in STD_LOGIC_VECTOR (3 downto 0);
           data1 : in STD_LOGIC_VECTOR (3 downto 0);
           data100 : in STD_LOGIC_VECTOR (3 downto 0);
           d1 : out STD_LOGIC_VECTOR (3 downto 0);
           d10 : out STD_LOGIC_VECTOR (3 downto 0);
           d100 : out STD_LOGIC_VECTOR (3 downto 0));
    end component;
    signal clk_tb,load_tb,reset_global_tb:STD_LOGIC;
    signal data1_tb,data10_tb,data100_tb:STD_LOGIC_VECTOR (3 downto 0);
    signal d1_tb,d10_tb,d100_tb:STD_LOGIC_VECTOR (3 downto 0);

begin
    UUT:clk_demical_counter port
map(reset_global=>reset_global_tb,clk=>clk_tb,load=>load_tb,data10=>data10_tb,
data1=>data1_tb,data100=>data100_tb,d1=>d1_tb,d10=>d10_tb,d100=>d100_tb);
    reset_global_tb<='1','0' after 160ns;
    load_tb<='0','1' after 160ns,'0' after 320ns;
    data1_tb<="0100";
    data10_tb<="1000";
    data100_tb<="0010";
```

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

For test bench, the value of `reset_global_tb` is set to 1 in the first $160ns$ and 0 after that duration. The value of `load_tb` is set to 0 in the first $160ns$, 1 after the second $160ns$ and 0 after these duration. The values of `data1_tb`, `data10_tb` and `data100_tb` are set to 0100, 1000 and 0010. The period of clock is $10ns$, which is the same as that of clock on FPGA board.

`Attention: because simulation in seconds takes a long time to compute using VIVADO, the period of output of frequency divider is set to 10μs. All of the following simulation is based on this.`

2.4 Behavioral simulation

The result of behavioral simulation with increasing 1 by 10μs is in `Fig 4`:

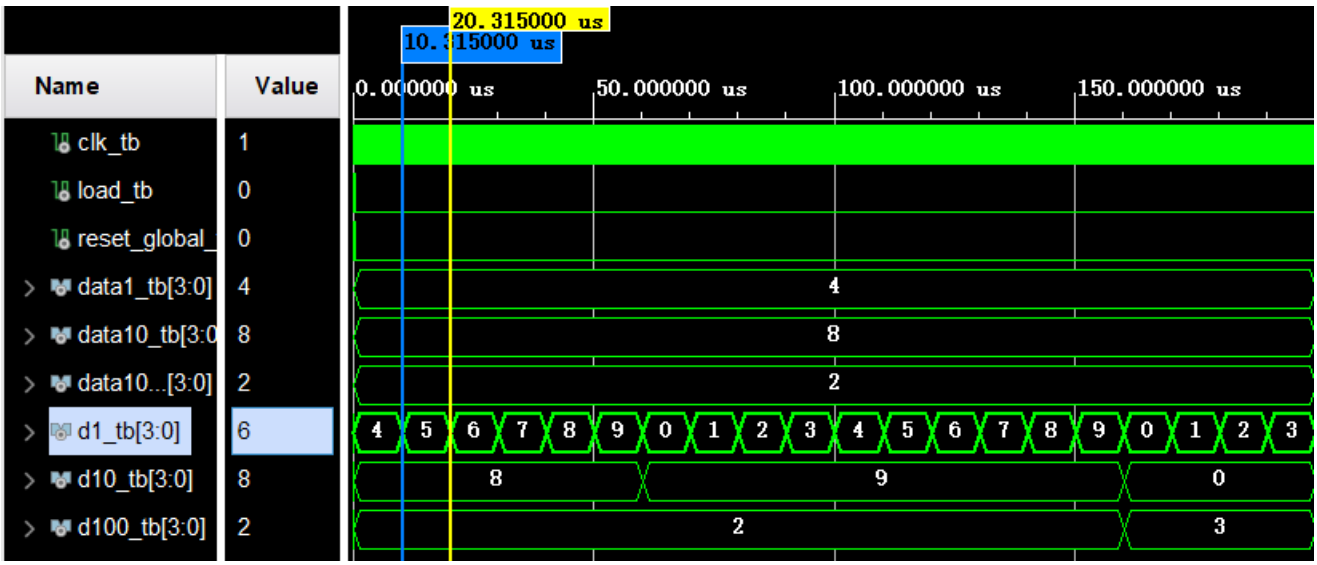
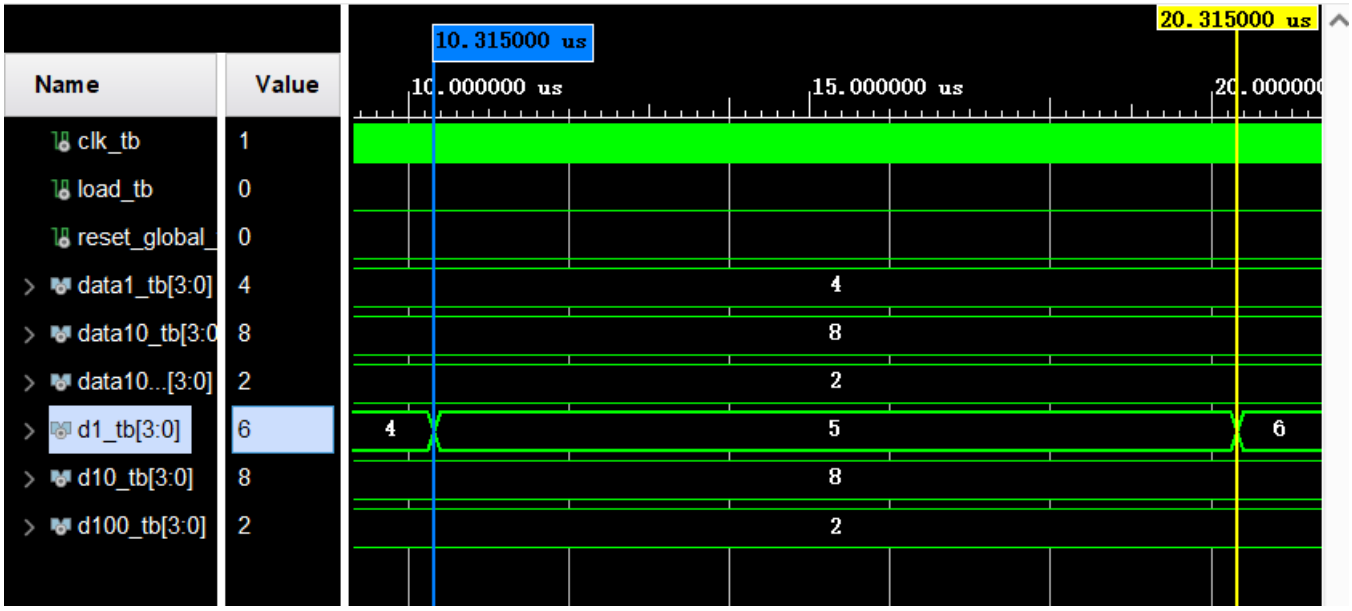
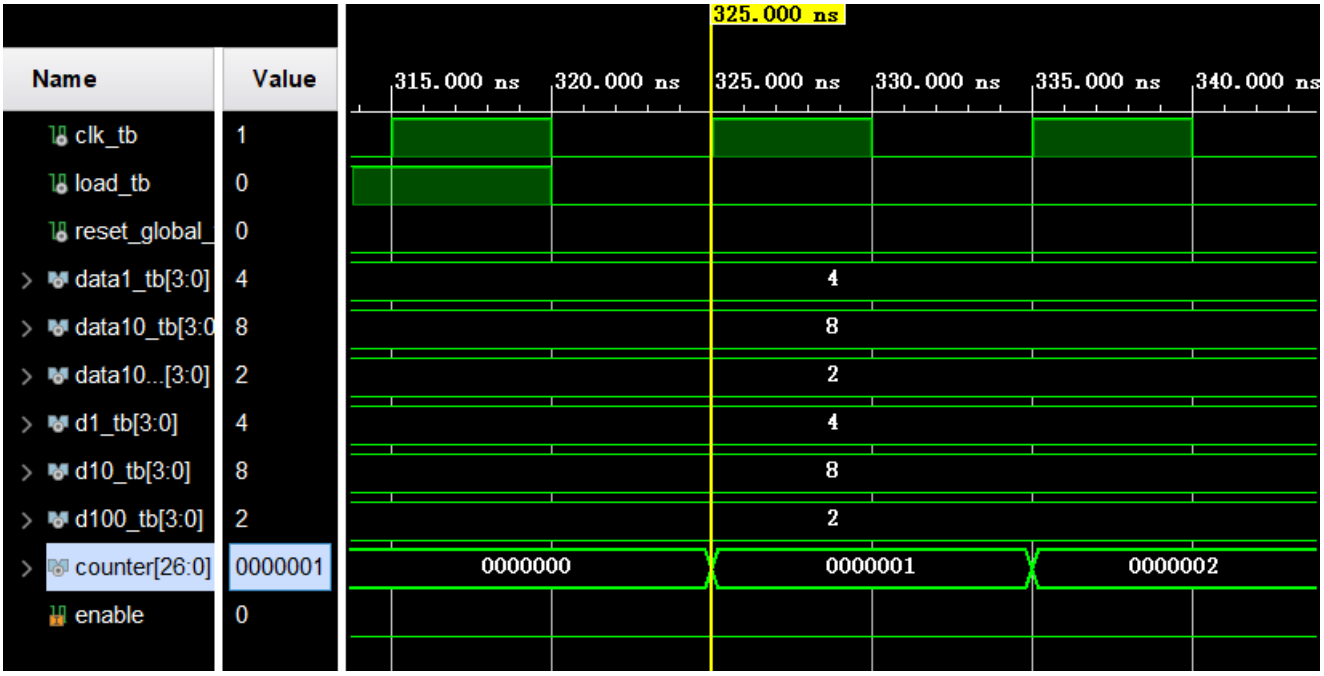


Fig.4 Behavioral simulation result with increasing 1 by 10μs


Name
Value
clk_tb
1
load_tb
0
reset_global_
0
data1_tb[3:0]
4
data10_tb[3:0
8
data10...[3:0]
2
d1_tb[3:0]
6
d10_tb[3:0]
8
d100_tb[3:0]
2
10.315000 us
20.315000 us
15.000000 us
4
8
2
4
5
6
8
2


Name
Value
clk_tb
1
load_tb
0
reset_global_
0
data1_tb[3:0]
4
data10_tb[3:0
8
data10...[3:0]
2
d1_tb[3:0]
4
d10_tb[3:0]
8
d100_tb[3:0]
2
counter[26:0]
0000001
enable
0
325.000 ns
315.000 ns
320.000 ns
325.000 ns
330.000 ns
335.000 ns
340.000 ns
4
8
2
4
8
2
0000000
0000001
0000002

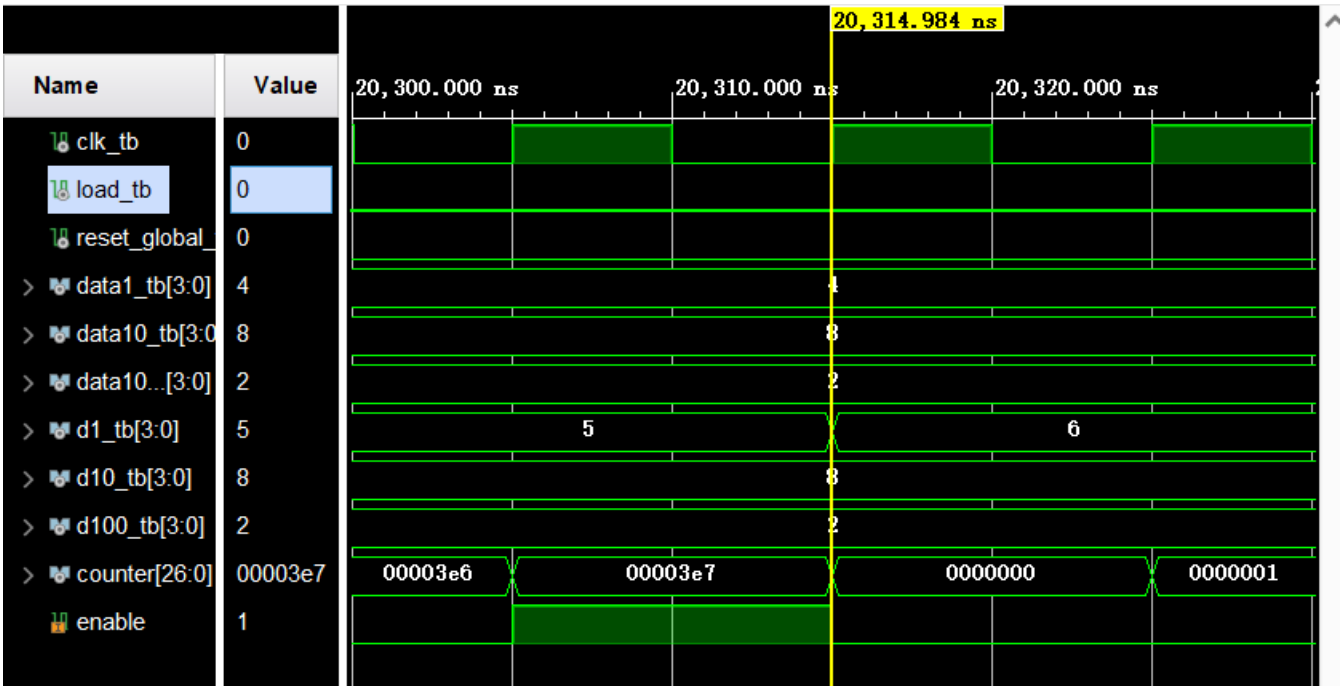


Fig.5 details of Behavioral simulation result with increasing 1 by 10μs

From the details of Behavioral simulation result with increasing 1 by 10μs, the result of simulation is in line with our expectation. When the logic value of `reset` is 1, the counter is initialized to 000, which satisfy asynchronous reset function. The load number are loaded when both of `clk` and `load` are 1, which satisfy synchronous load function. The counter of frequency divider start counting at the first rising edge of `clk` after the logic value of `load` is 0. The whole decimal counter increase by 1 after 10μs. What's more, the duration of high level of `enable` is the period of `clk`, which ensures the whole decimal counter increase by 1 after 10μs.

Then we test the simulation when the value in the corresponding position is more than 9. The number which will loaded is 001011000100. The result of simulation is as below:

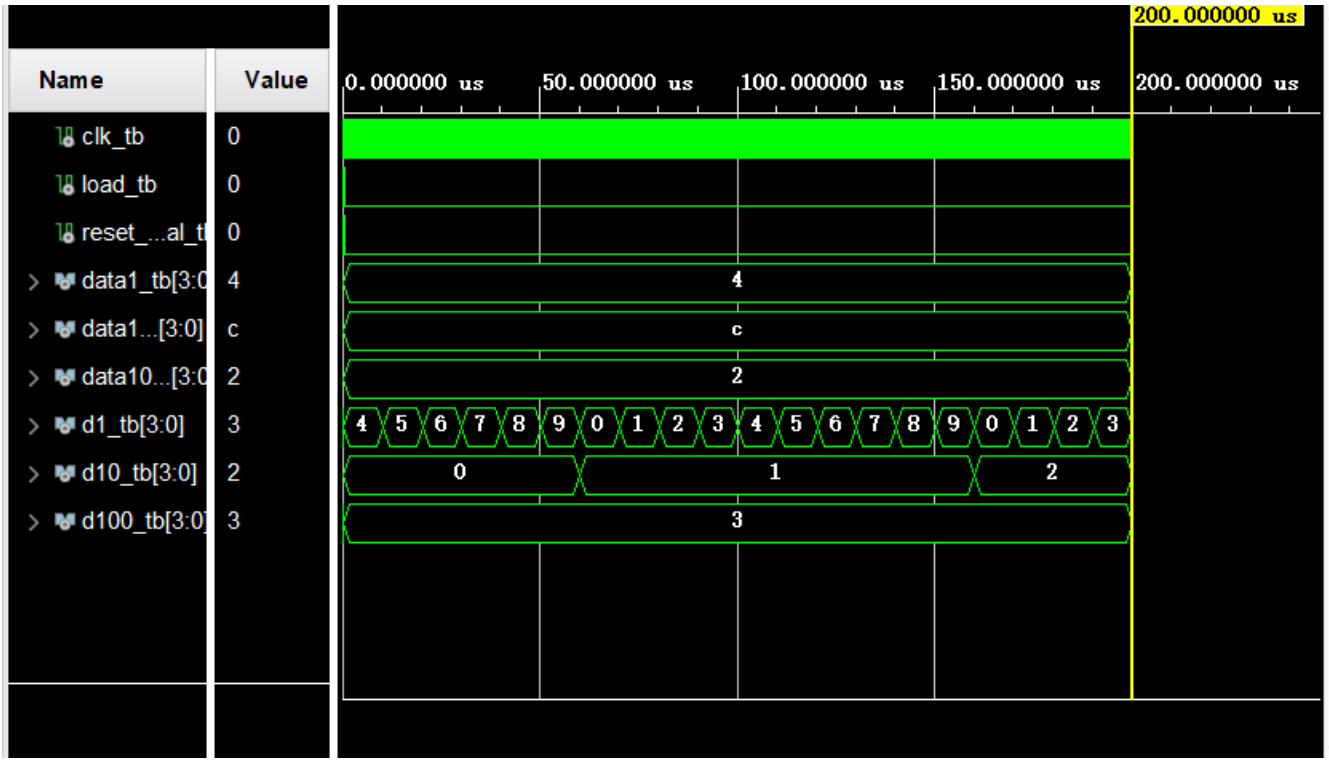


Fig.6 Behavioral simulation result for the value in the corresponding position is more than 9

The loaded number is 304, which is in line with our expectation.

2.5 Post-Synthesis simulation

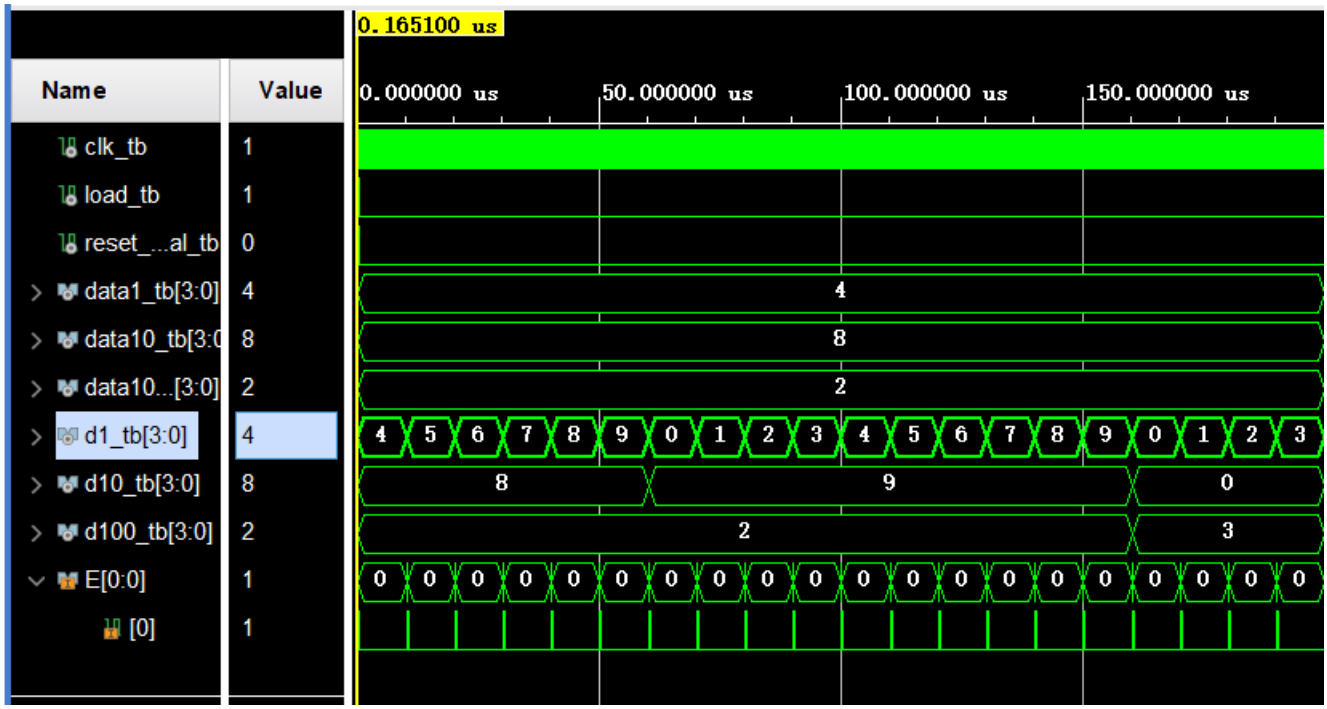
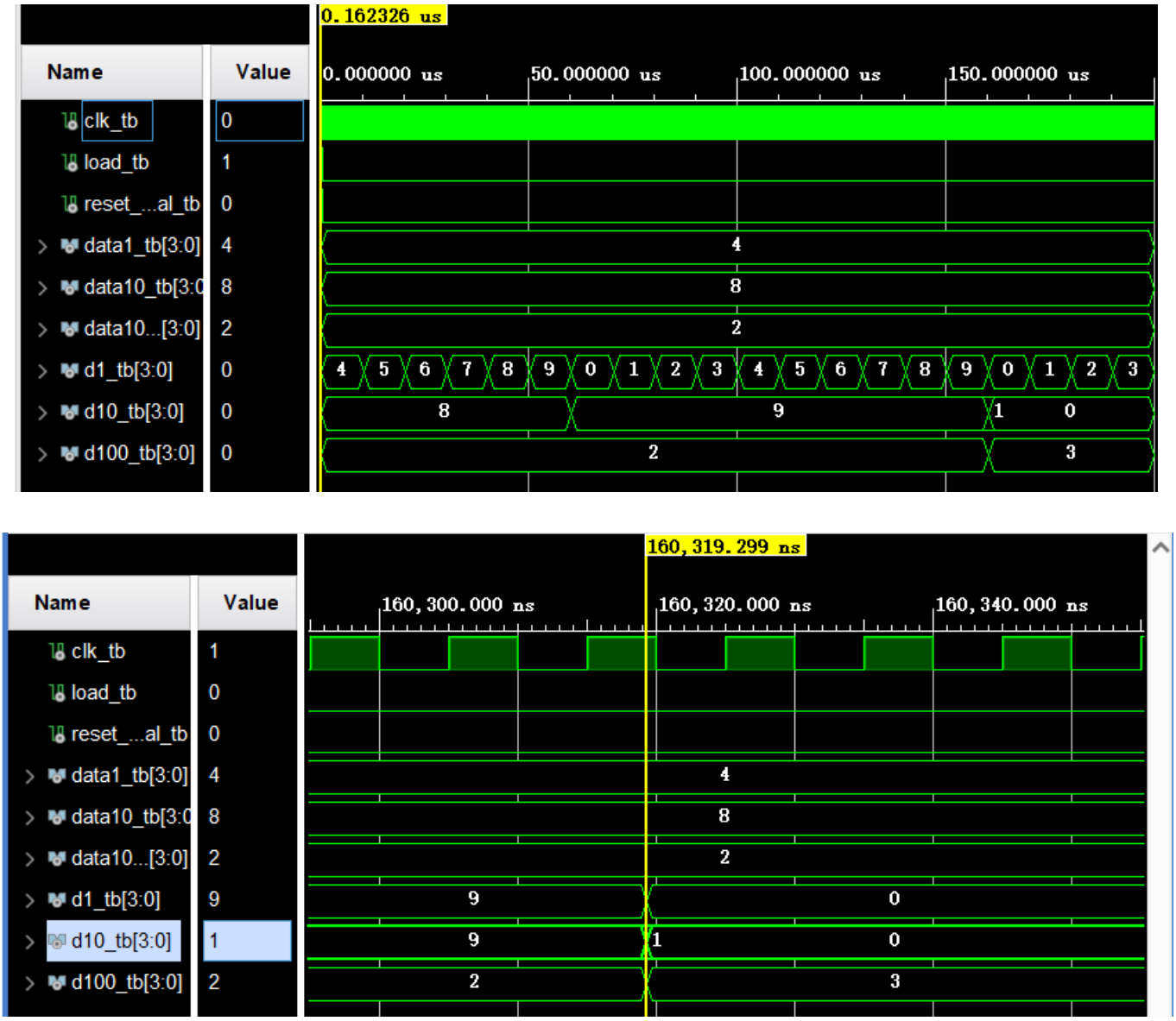


Fig.7 Post-Synthesis functional simulation result with increasing 1 by 10μs

The post-synthesis functional simulation is also in line with our design and the result of behavioral simulation. However, the result of post-synthesis timing simulation is a little different from that of behavioral simulation.



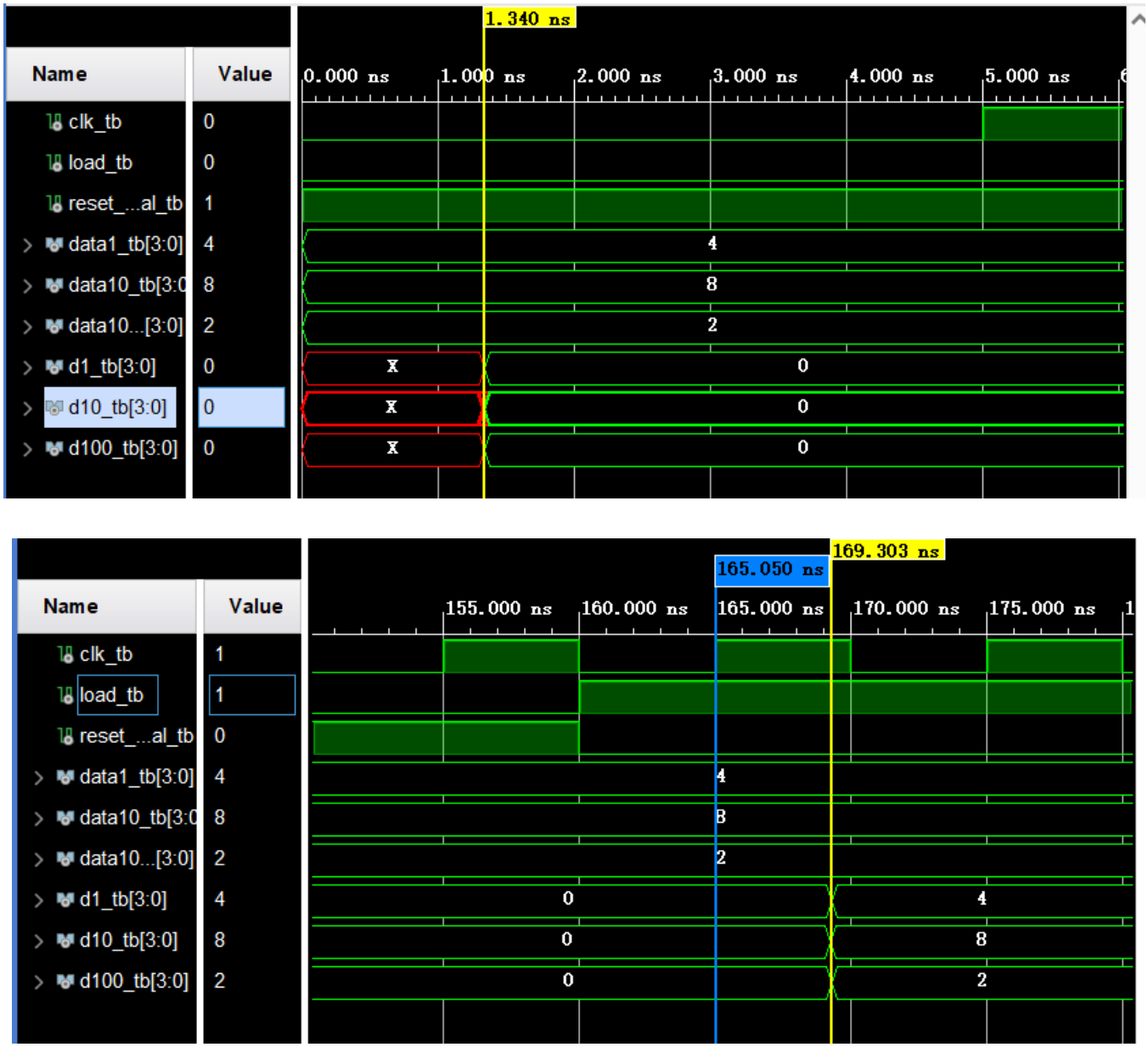


Fig.8 Post-Synthesis timing simulation result with increasing 1 by 10μs

It is obviously that there exists a delta delay in the beginning, which is different from behavioral simulation and postsynthesis functional simulation. The value of delta delay is 1.340 ns. What's more, the numbers of input are not loaded until about 4.303ns after the rising edge of `load`. The reason behind this is that although multiple signal assignment statements are executed concurrently in simulated time and are referred to as concurrent signal assignment statements, a small delay——delta delay will be automatically set in order to make a logical sequence of signals transmission.

As we can see, there is instantaneous change in `d10_tb`. One possible reason is that adding 1 in the combinational circuit when the output is 299(decimal), `d1_tb` and `d10_tb` should change from 9 to 0. The binary code for 9 is 1001. The signal that changes the highest bit from 1 to 0 arrives before the signal that changes the lowest bit from 1 to 0. When the highest bit changes from 1 to 0, the value of the lowest bit is still 1 in that very short period of time. That's why there is instantaneous change in `d10_tb`.

2.6 Post-Synthesis simulation

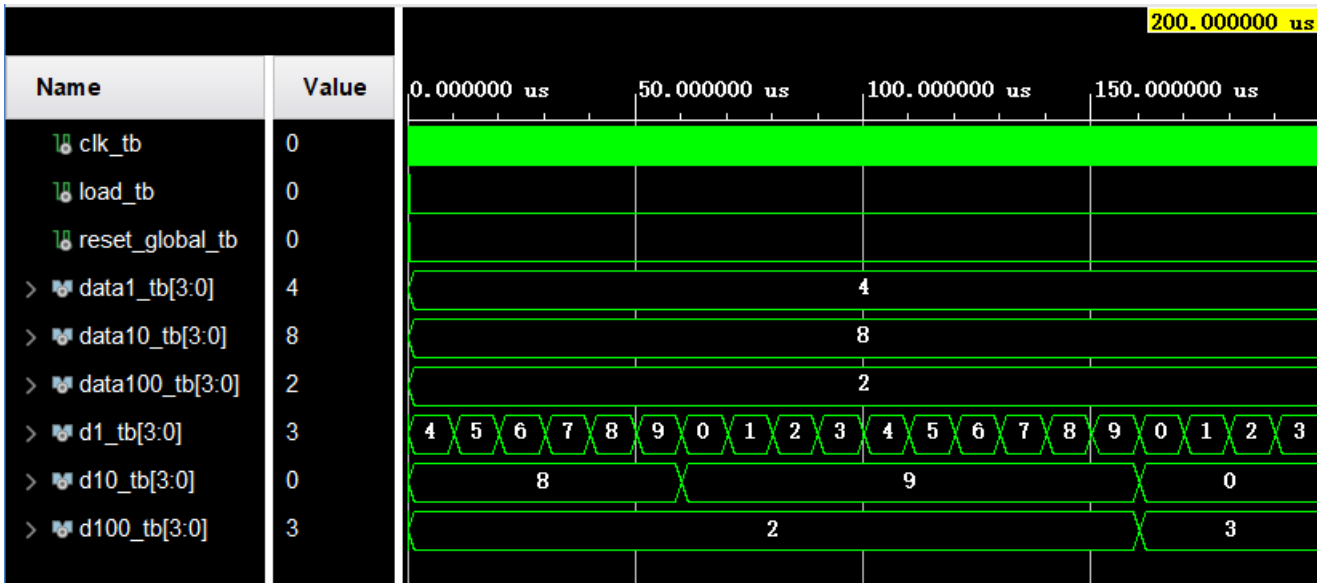
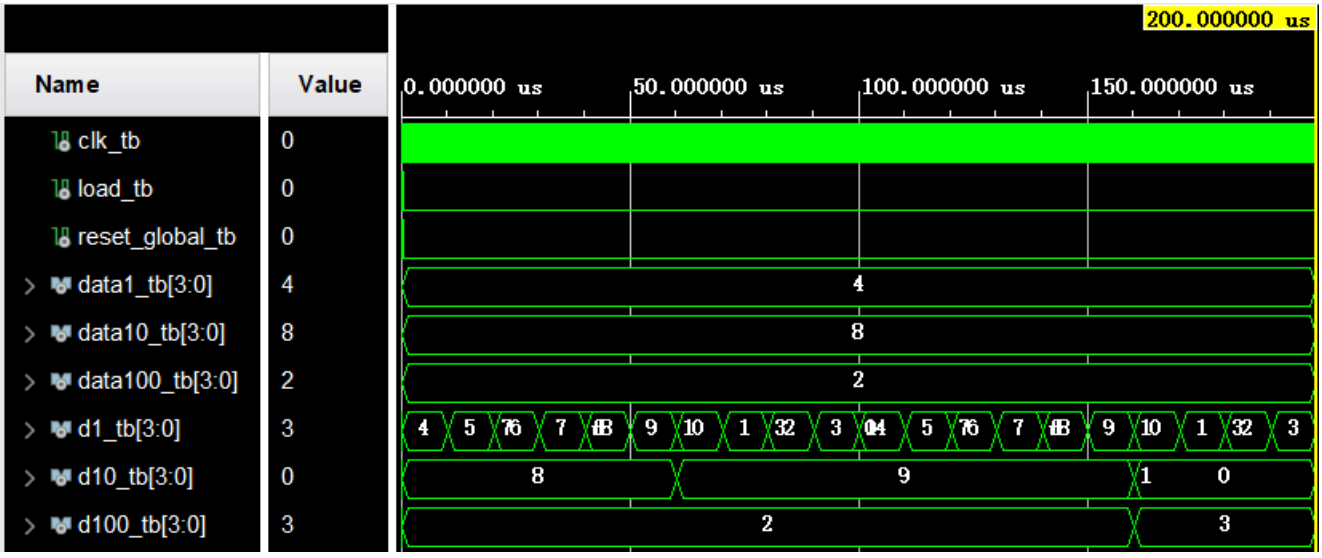
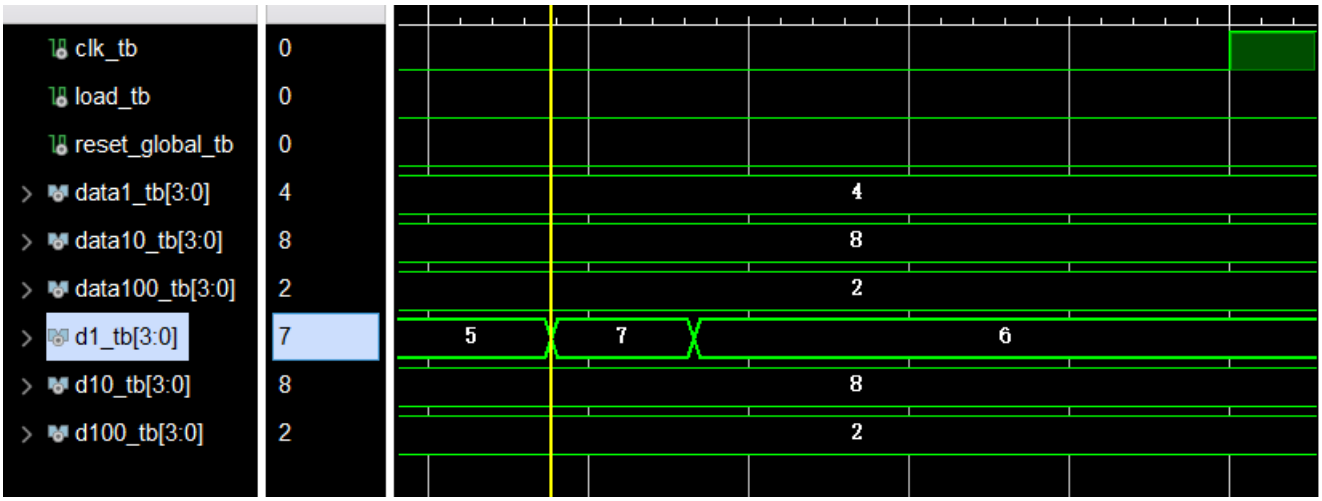
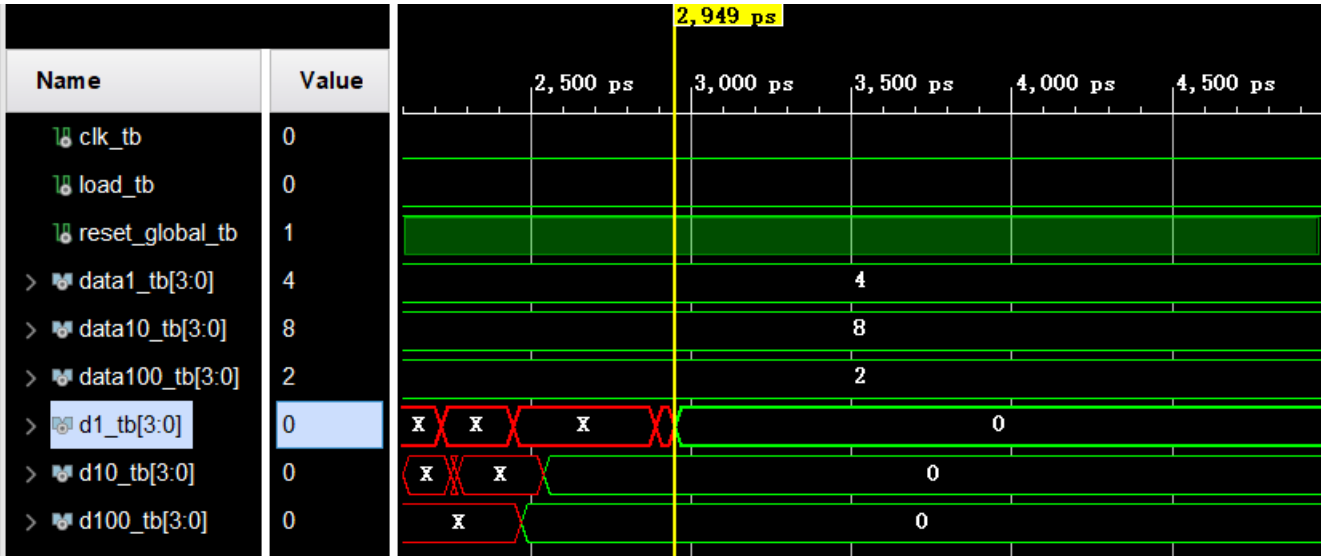


Fig.9 Post-implementtation functional simulation result with increasing 1 by 10μs

The post-implementation functional simulation is also in line with our design and the result of behavioral simulation.


Name
Value
clk_tb
0
load_tb
0
reset_global_tb
0
data1_tb[3:0]
4
data10_tb[3:0]
8
data100_tb[3:0]
2
d1_tb[3:0]
3
d10_tb[3:0]
0
d100_tb[3:0]
3
200.000000 us
0.000000 us
50.000000 us
100.000000 us
150.000000 us


clk_tb
0
load_tb
0
reset_global_tb
0
data1_tb[3:0]
4
data10_tb[3:0]
8
data100_tb[3:0]
2
d1_tb[3:0]
7
d10_tb[3:0]
8
d100_tb[3:0]
2


Name
Value
clk_tb
0
load_tb
0
reset_global_tb
1
data1_tb[3:0]
4
data10_tb[3:0]
8
data100_tb[3:0]
2
d1_tb[3:0]
0
d10_tb[3:0]
0
d100_tb[3:0]
0
2,949 ps
2,500 ps
3,000 ps
3,500 ps
4,000 ps
4,500 ps

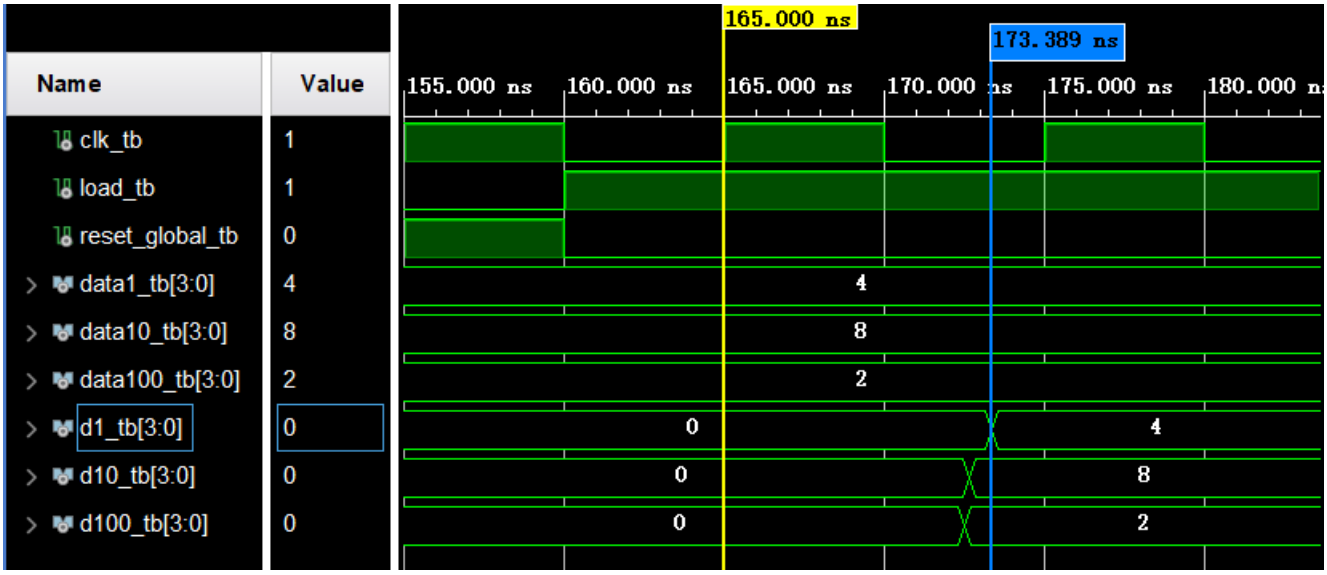


Fig.10 Post-implementtation timing simulation result with increasing 1 by 10μs

It is obviously that the delays at the beginning of `d1_tb`, `d10_tb` and `d100_tb` are different, which are almost twice as that in post-implementation timing simulation. What's more, the number of input `data1_tb` are not loaded until about 8.389ns after the rising edge of `load`, which are also almost twice as that in post-implementation timing simulation. That's due to layout, routing and different time delay of gate circuits in post-implementation timing simulation.

The schematic is very similar to our block diagram, which verify our code’s correctness.

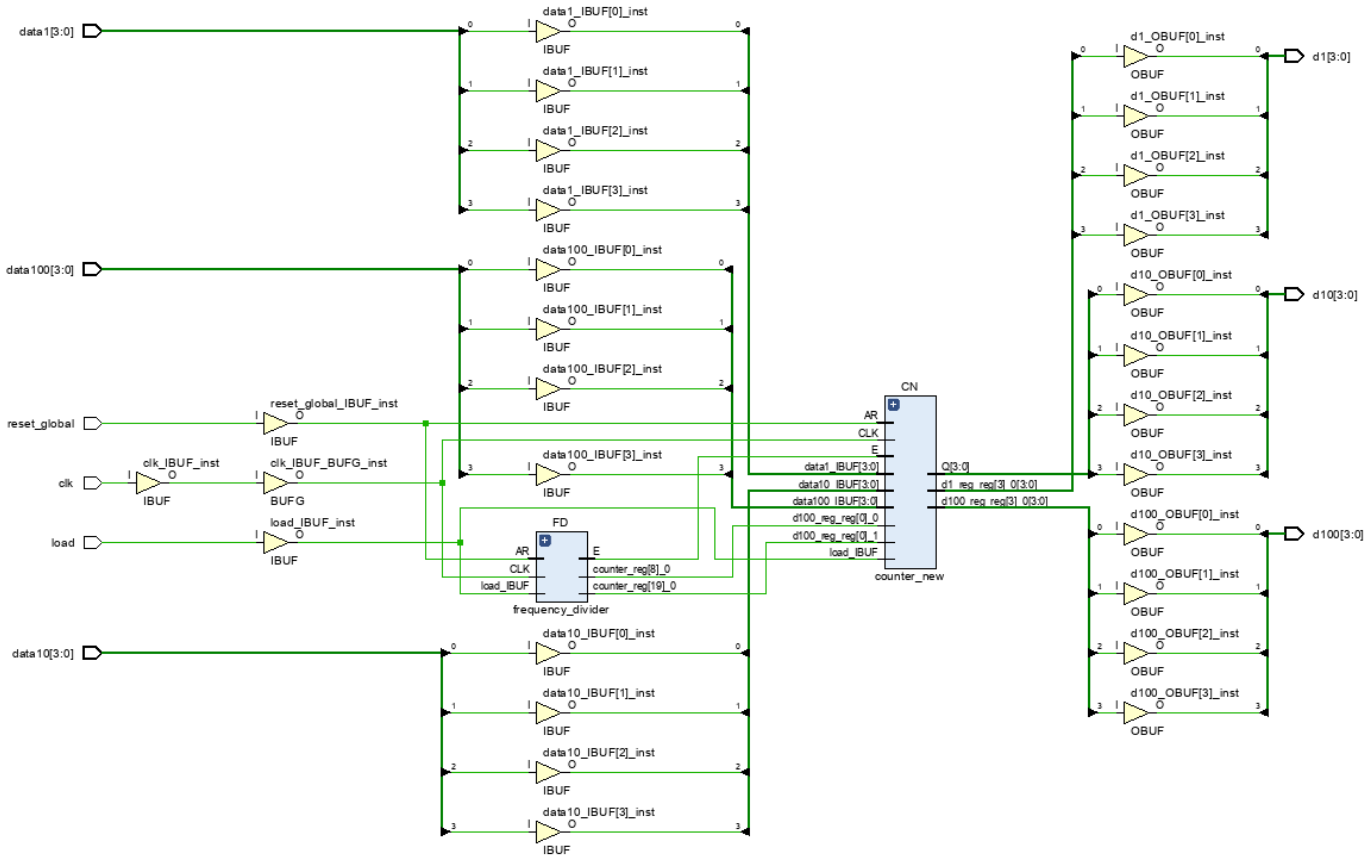


Fig.11 schematic

2.7 On board verify

In order to burn the program to FPGA, we need to write constraints file. The code for constraints file is as below:

```
set_property PACKAGE_PIN E3 [get_ports clk]
set_property PACKAGE_PIN M17 [get_ports reset_global]
set_property PACKAGE_PIN V10 [get_ports load]

set_property PACKAGE_PIN J15 [get_ports data1[0]]
set_property PACKAGE_PIN L16 [get_ports data1[1]]
set_property PACKAGE_PIN M13 [get_ports data1[2]]
set_property PACKAGE_PIN R15 [get_ports data1[3]]

set_property PACKAGE_PIN R17 [get_ports data10[0]]
set_property PACKAGE_PIN T18 [get_ports data10[1]]
```

```
set_property PACKAGE_PIN U18 [get_ports data10[2]]
set_property PACKAGE_PIN R13 [get_ports data10[3]]

set_property PACKAGE_PIN T8 [get_ports data100[0]]
set_property PACKAGE_PIN U8 [get_ports data100[1]]
set_property PACKAGE_PIN R16 [get_ports data100[2]]
set_property PACKAGE_PIN T13 [get_ports data100[3]]

set_property PACKAGE_PIN H17 [get_ports d1[0]]
set_property PACKAGE_PIN K15 [get_ports d1[1]]
set_property PACKAGE_PIN J13 [get_ports d1[2]]
set_property PACKAGE_PIN N14 [get_ports d1[3]]

set_property PACKAGE_PIN R18 [get_ports d10[0]]
set_property PACKAGE_PIN V17 [get_ports d10[1]]
set_property PACKAGE_PIN U17 [get_ports d10[2]]
set_property PACKAGE_PIN U16 [get_ports d10[3]]

set_property PACKAGE_PIN V16 [get_ports d100[0]]
set_property PACKAGE_PIN T15 [get_ports d100[1]]
set_property PACKAGE_PIN U14 [get_ports d100[2]]
set_property PACKAGE_PIN T16 [get_ports d100[3]]
```

The layout is shown in `Fig 12`.

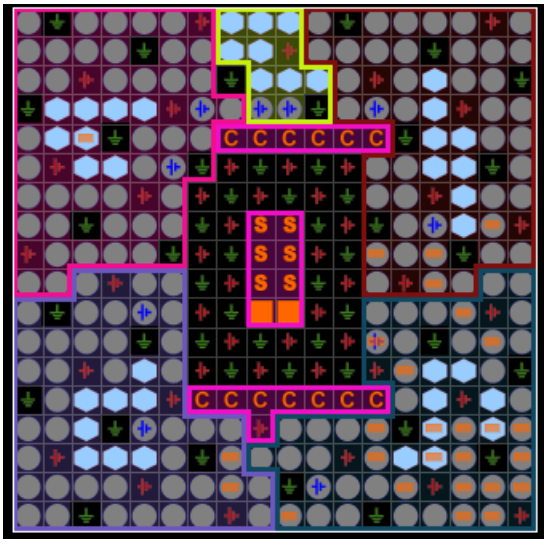


Fig.12 layout

And finally, we set the frequency to 1Hz and generate the bitstream and program to the board, the decimal counter result is correct. The result is shown in `Fig 13`.

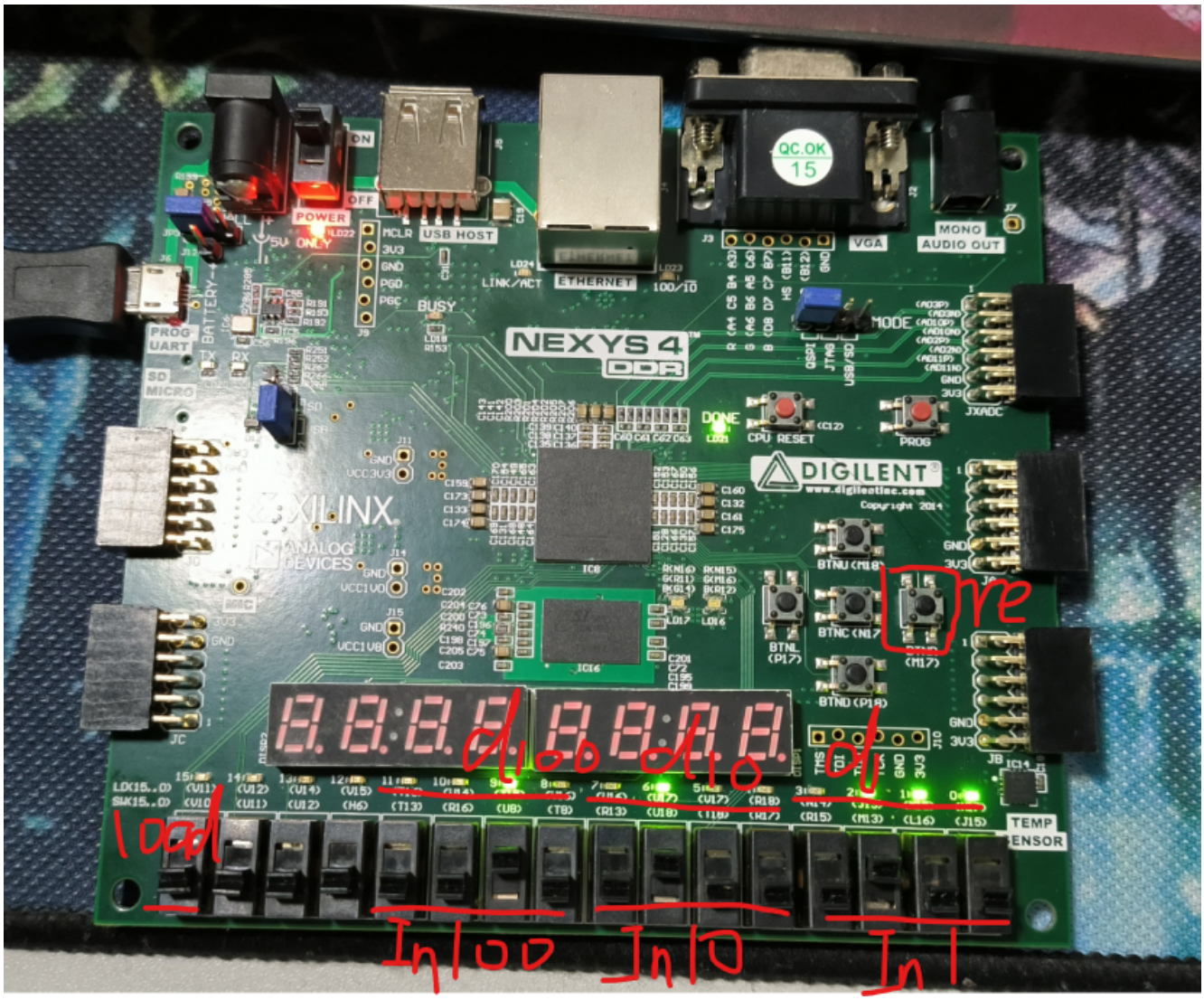


Fig.13 Test on board

3.Conclusion

The main part of this report is the process of digital design of three-digit decimal counter using VIVADO. There are some important points that need to be stressed again:

1. Synchronization means under the excitation of clock signal, in the rising or falling edge of the clock, the trigger carries out the corresponding operation.

2. Asynchrony means that the trigger performs corresponding operation at the rising or falling edge of the signal under the excitation signal.
3. The first segment of two-segment coding style is for a synchronous section or a synchronous section with asynchronous inputs. The first segment of two-segment coding style is for a combinational section.