计数器的设计实验报告

翁家翌 2016011446

2018.5.11

1 实验目的

1. 掌握时序电路的基本分析和设计方法
2. 理解同步时序电路和异步时序电路的区别
3. 掌握计数器电路的设计原理，用硬件描述语言实现指定功能的计数器设计
4. 学会利用软件仿真实现对数字电路的逻辑功能进行验证分析

2 实验内容

1. 使用实验平台上两个未经译码处理的数码管显示计数，手动单次时钟进行计数，时钟上升沿计数一次，当计数到 59 的时候，要求两个数码管都能复位到 00 的状态，重新计数，实验还要求设置一个复位键，可以随时重新恢复到 00 的状态继续计数。
2. 使用实验平台上的 1MHz 时钟，将计数器改成秒表，在秒表中使用开关控制秒表启动暂停。

3 代码及注释

3.1 触发器元件定义

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity ff is
7          port(
8                  clk, rst, mode, pause: in std_logic;
9                  n0, n1: buffer std_logic_vector(3 downto 0)
10         );
11 end ff;
12 architecture arc of ff is
13         signal cnt: integer := 0;
```

```
14 begin
15         process(clk, rst)
16         begin
17                 if (rst = '0') then
18                         n0 <= "0000";
19                         n1 <= "0000";
20                         cnt <= 0;
21                 elsif (clk'event and clk = '1' and pause = '0')
                       then
22                         if (mode = '1') then -- clocker (Task#2)
23                                 if (cnt < 1000000) then
24                                         cnt <= cnt + 1;
25                                 else
26                                         cnt <= 0;
27                                 end if;
28                         end if;
29                         if (mode = '0' or cnt = 0) then -- +1
30                                 if (n0 = "1001") then
31                                         n0 <= "0000";
32                                         if (n1 = "0101") then
33                                                 n1 <= "0000";
34                                         else
35                                                 n1 <= n1 + 1;
36                                         end if;
37                                 else
38                                         n0 <= n0 + 1;
39                                 end if;
40                         end if;
41                 end if;
42         end process;
43 end arc;
```

3.2 二进制转非译码器显示元件定义

```
1 entity digit_7 is
2         port(
3                 number: in std_logic_vector(3 downto 0);
4                 display: out std_logic_vector(6 downto 0)
5         );
6 end digit_7;
7 architecture arc of digit_7 is
8 begin
```

```
9         process(number)
10        begin
11                case number is
12                        when "0000"=>display<="1111110";
13                        when "0001"=>display<="0110000";
14                        when "0010"=>display<="1101101";
15                        when "0011"=>display<="1111001";
16                        when "0100"=>display<="0110011";
17                        when "0101"=>display<="1011011";
18                        when "0110"=>display<="0011111";
19                        when "0111"=>display<="1110000";
20                        when "1000"=>display<="1111111";
21                        when "1001"=>display<="1110011";
22                        when "1010"=>display<="1110111";
23                        when "1011"=>display<="0011111";
24                        when "1100"=>display<="1001110";
25                        when "1101"=>display<="0111101";
26                        when "1110"=>display<="1001111";
27                        when "1111"=>display<="1000111";
28                        when others=>display<="0000000";
29                end case;
30        end process;
31 end arc;
```

3.3 程序主体

```
1  entity count is
2         port(
3                 clk, rst, mode, pause: in std_logic;
4                 n0, n1: buffer std_logic_vector(3 downto 0);
5                 ll, hh: out std_logic_vector(6 downto 0)
6         );
7  end count;
8
9  architecture arc of count is
10        component ff
11                port(
12                        clk, rst, mode, pause: in std_logic;
13                        n0, n1: buffer std_logic_vector(3 downto
                              0)
14                );
15        end component;
```

```
16          component digit_7
17                  port(
18                          number: in std_logic_vector(3 downto 0);
19                          display: out std_logic_vector(6 downto 0)
20                  );
21          end component;
22  begin
23          tmp0: ff port map(clk=>clk, rst=>rst, n0=>n0, n1=>n1,
                mode=>mode, pause=>pause);
24          tmp1: digit_7 port map(number=>n0, display=>ll);
25          tmp2: digit_7 port map(number=>n1, display=>hh);
26  end arc;
```

工作原理：采用元件例化的方法，构造一个将二进制转化为一个可以在不带译码器的数码管上显示计数，这部分代码可以直接采用第一次写 ``点亮数字人生'' 的代码，这样两位数字显示只需要写一位数字显示的代码，提高了代码复用率。

另一个元件是触发器，当遇到时钟上升沿的时候，计数器 +1（如果是时钟模式则是计数信号 +1，计数信号满 1M 之后计数器 +1）。对于 rst 信号，将存储的数值清零即可，时钟模式时也把计数信号清零。

4 仿真结果

如图1所示。

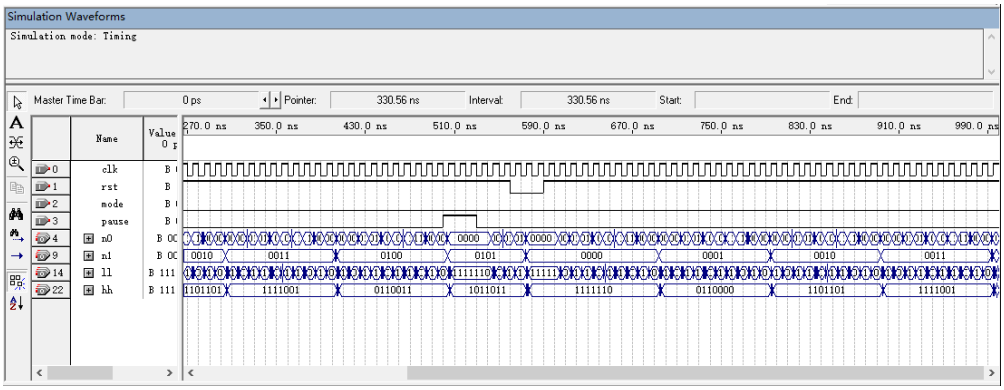

图 1: 仿真结果，包含进位、暂停与清零的效果

5 实验小结

这是我第三次进行 CPLD 实验，在之前的实验的基础上又有了不小的提高，可以看懂出现的比较常见的 Bug 提示，而且能根据提示进行对应的 Debug。

通过本次实验，对于触发器的理解更加透彻，也能够更轻松地利用时钟信号来完成任务。在完成此次代码的时候，还利用了一部分之前写过的代码，也复习了之前学的知识，温故知新。