Lab 2 Exercise 4:

Design and realize a three-digit decimal counter

姓名：王宇　　　学号：12112725

原理图：

分频器采用的设计如图所示，因 100MHZ 的时钟频率对应的是 10ns，与 1s 相差10^8，转换为二进制共有27位，故c_next和c_reg均为27位。当c_reg的值不等于10^8-1 时，c_next+1，当 c_reg 的值等于10^8-1 时，即过去 1s 时，c_next 置为 0，重新开始计时。

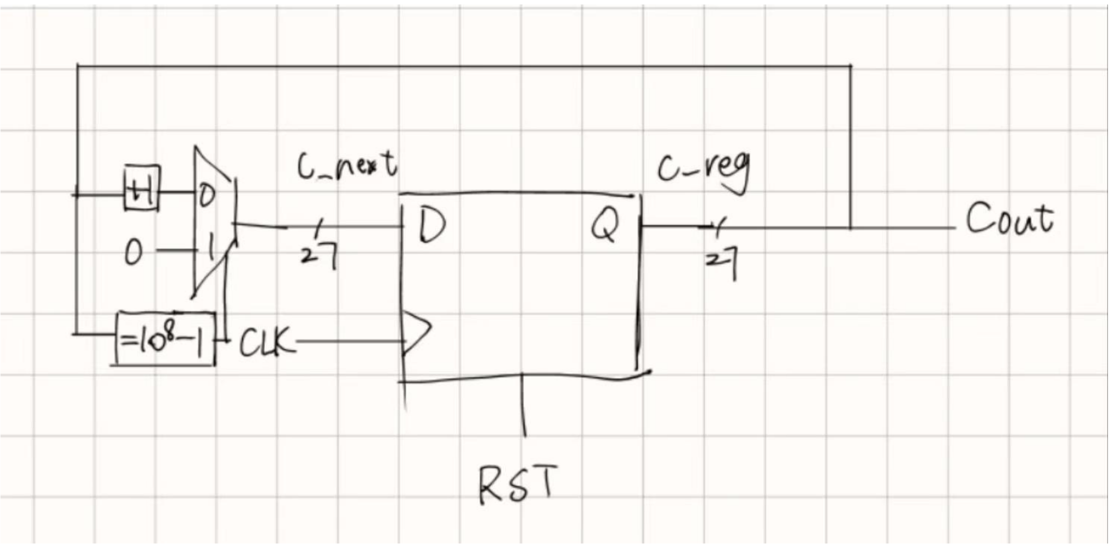

图 1 frequency divider 概念图

三位计数器采用的设计如图所示，当 d1_reg 不等于 9 时，则+1，等于 9 时置 0，并且传递到下一位的判断，若 d2_reg 不等于 9 且 d1_reg 等于 9 时+1，二者均等于 9 时，d2_reg 置 0，其余逻辑均类似。

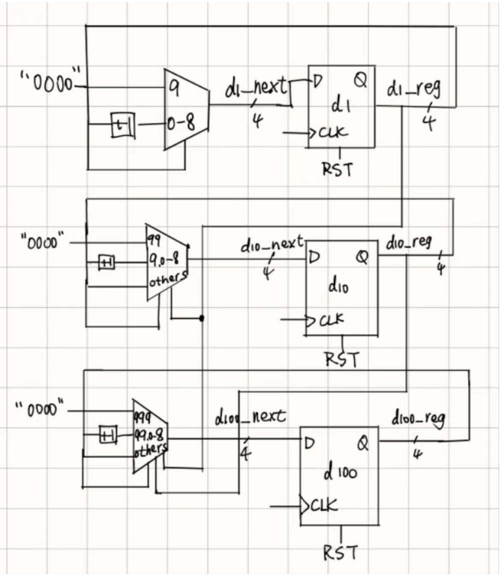

图 2

3-digit decimal counter 概念图

总体设计如下图所示，在分频器的输出端加了一个判断条件，并在 d1 上加了一个使能端 EN。若分频器输出为10^8-1，则使能端置为 1，计数器开始工作，否则停止工作。图中所有时钟信号均为同一时钟信号。

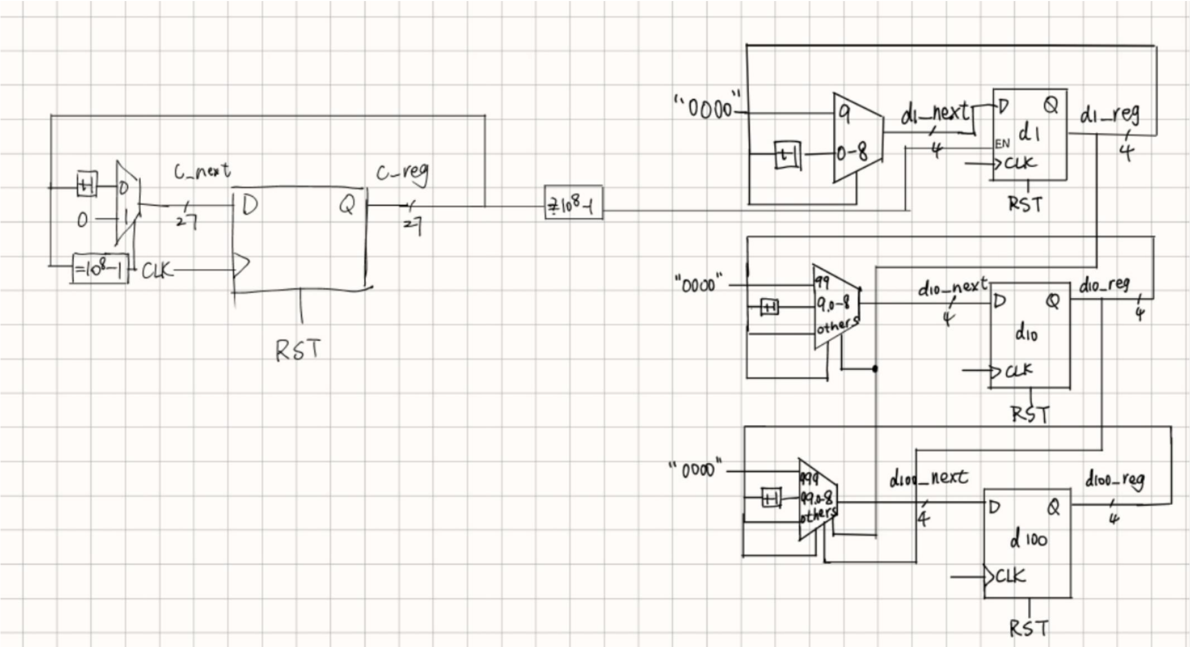

图 3 top level 结构图

代码：

VHDL 部分：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity decimal_counter is
    Port ( CLOCK : in STD_LOGIC;
           RESET : in STD_LOGIC;
           SW : in STD_LOGIC_VECTOR(11 downto 0);
           d1, d10, d100: out STD_LOGIC_VECTOR(3 downto 0));
end decimal_counter;

architecture Behavioral of decimal_counter is
    signal d1_reg, d10_reg, d100_reg: STD_LOGIC_VECTOR(3 downto 0);
    signal c_reg: STD_LOGIC_VECTOR(26 downto 0);
    signal bin_value : STD_LOGIC_VECTOR(11 downto 0);
    signal EN: STD_LOGIC;

begin
    -- 时钟部分
    process(CLOCK, RESET)
```

```
begin
    if RESET = '0' then
        c_reg <= (others => '0');
        EN <= '0';
    elsif rising_edge(CLOCK) then
        if c_reg = "1011111010111110000100000000" then -- Replace with your condition
            c_reg <= (others => '0');
            EN <= '1';
        else
            c_reg <= std_logic_vector(unsigned(c_reg) + 1);
            EN <= '0';
        end if;
    end if;
end process;

-- 计数器部分
process(CLOCK, RESET)
begin
    if RESET = '0' then
        d1_reg <= SW(3 downto 0);
        d10_reg <= SW(7 downto 4);
        d100_reg <= SW(11 downto 8);
    elsif rising_edge(CLOCK) then
        if EN = '1' then
            if unsigned(d1_reg) = 9 then
                d1_reg <= "0000";
                if unsigned(d10_reg) = 9 then
                    d10_reg <= "0000";
                    if unsigned(d100_reg) = 9 then
                        d100_reg <= "0000";
                    else
                        d100_reg <= std_logic_vector(unsigned(d100_reg) + 1);
                    end if;
                else
                    d10_reg <= std_logic_vector(unsigned(d10_reg) + 1);
                end if;
            else
                d1_reg <= std_logic_vector(unsigned(d1_reg) + 1);
            end if;
        end if;
    end if;
end process;

-- 输出部分
```

```
    d1 <= d1_reg;
    d10 <= d10_reg;
    d100 <= d100_reg;
end Behavioral;
```

Testbench 部分：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

ENTITY tb_decimal_counter IS
END tb_decimal_counter;

ARCHITECTURE behavior OF tb_decimal_counter IS
    COMPONENT decimal_counter
    PORT(
         CLOCK : IN    std_logic;
         RESET : IN    std_logic;
         SW : IN STD_LOGIC_VECTOR(11 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 CLOCK : std_logic := '0';
    signal RESET : std_logic := '0';
    signal SW : std_logic_vector(11 downto 0) := (others => '0');
    signal d1 : std_logic_vector(3 downto 0);
    signal d10 : std_logic_vector(3 downto 0);
    signal d100 : std_logic_vector(3 downto 0);

    constant CLOCK_PERIOD : time := 10 ns; -- 100MHz 时钟

BEGIN
    uut: decimal_counter PORT MAP (
          CLOCK => CLOCK,
          RESET => RESET,
          SW => SW,
          d1 => d1,
          d10 => d10,
          d100 => d100
        );
    CLOCK_process :process
    begin
        CLOCK <= '0';
        wait for CLOCK_PERIOD/2;
        CLOCK <= '1';
        wait for CLOCK_PERIOD/2;
    end process;

    stim_proc: process
    begin
        RESET <= '1';
        wait for 1000 ns;
        RESET <= '0';
        wait for 1000 ns;
        SW <= "000000000001";

        wait for 5000 ms;
    end process;

END;
```

约束条件部分：

```
# Reset button (suppose the button you wish to use for reset is BTN0)
set_property PACKAGE_PIN C12 [get_ports {RESET}]
set_property IOSTANDARD LVCMOS33 [get_ports {RESET}]

# Clock signal
set_property PACKAGE_PIN E3 [get_ports {CLOCK}]
set_property IOSTANDARD LVCMOS33 [get_ports {CLOCK}]
create_clock -add -name sys_clk_pin -period 10.000 -waveform {0 5} [get_ports {CLOCK}]

# 12 switches for setting the start number (suppose SW1 - SW12 are used)
set_property PACKAGE_PIN V10 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[0]}]
set_property PACKAGE_PIN U11 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[1]}]
set_property PACKAGE_PIN U12 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[2]}]
set_property PACKAGE_PIN H6 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[3]}]
set_property PACKAGE_PIN T13 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[4]}]
set_property PACKAGE_PIN R16 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[5]}]
set_property PACKAGE_PIN U8 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[6]}]
set_property PACKAGE_PIN T8 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[7]}]
set_property PACKAGE_PIN R13 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[8]}]
set_property PACKAGE_PIN U18 [get_ports {SW[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[9]}]
set_property PACKAGE_PIN T18 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[10]}]
set_property PACKAGE_PIN R17 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW[11]}]

# 12 LEDs for showing
set_property PACKAGE_PIN R18 [get_ports { d1[0] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d1[0]}]
set_property PACKAGE_PIN V17 [get_ports { d1[1] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d1[1]}]
set_property PACKAGE_PIN U17 [get_ports { d1[2] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d1[2]}]
set_property PACKAGE_PIN U16 [get_ports { d1[3] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d1[3]}]

set_property PACKAGE_PIN V16 [get_ports { d10[0] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d10[0]}]
set_property PACKAGE_PIN T15 [get_ports { d10[1] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d10[1]}]
set_property PACKAGE_PIN U14 [get_ports { d10[2] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d10[2]}]
set_property PACKAGE_PIN T16 [get_ports { d10[3] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d10[3]}]

set_property PACKAGE_PIN V15 [get_ports { d100[0] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d100[0]}]
set_property PACKAGE_PIN V14 [get_ports { d100[1] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d100[1]}]
set_property PACKAGE_PIN V12 [get_ports { d100[2] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d100[2]}]
set_property PACKAGE_PIN V11 [get_ports { d100[3] }]
set_property IOSTANDARD LVCMOS33 [get_ports {d100[3]}]
```