计数器的设计

计01　容逸朗　2020010869

1　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。

2　实验原理

2.1　D 触发器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

entity dtrigger is
  port(
    cp, rst, d: in std_logic;
    q, nq: buffer std_logic
  );
end dtrigger;

architecture bhv of dtrigger is
begin
  process(cp, rst)
  begin
    if (rst = '1') then
      q <= '0';
      nq <= '1';
    elsif (cp'event and cp = '1') then
      q <= d;
      nq <= not d;
    end if;
  end process;
end bhv;
```

工作原理：对于任意输入 D，若 CP 处于上升沿则有 $Q=D, \bar{Q}=\bar{D}$。若复位键被点击，则回到 $D=0$，即 $Q=0, \bar{Q}=1$ 的状态。

2.2 译码器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

entity digit7 is
  port(
    key: in std_logic_vector(3 downto 0);
    output: out std_logic_vector(6 downto 0)
  );
end digit7;

architecture bhv of digit7 is
begin
  process(key) --不带译码器的需要进行译码处理
  begin
    case key is --以下是编码规则
      when"0000"=> output<="1111110"; --0
      when"0001"=> output<="0110000"; --1
      when"0010"=> output<="1101101"; --2
      when"0011"=> output<="1111001"; --3
      when"0100"=> output<="0110011"; --4
      when"0101"=> output<="1011011"; --5
      when"0110"=> output<="1011111"; --6
      when"0111"=> output<="1110000"; --7
      when"1000"=> output<="1111111"; --8
      when"1001"=> output<="1110011"; --9
      when"1010"=> output<="1110111"; --a
      when"1011"=> output<="0011111"; --b
      when"1100"=> output<="1001110"; --c
      when"1101"=> output<="0111101"; --d
      when"1110"=> output<="1001111"; --e
      when"1111"=> output<="1000111"; --f
      when others=>output<="0000000"; --其他情况全灭
    end case;
  end process;
end bhv;
```

工作原理：将一个四位二进制数译为对应的七位数码管输出。

2.3 十位计数器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter10 is
port(
  clk, rst: in std_logic;
  res: buffer std_logic_vector(3 downto 0);
  car: buffer std_logic
);
end counter10;

architecture bhv of counter10 is
  component dtrigger is
    port(
      cp, rst, d: in std_logic;
      q, nq: buffer std_logic
    );
  end component;
  signal cnt: std_logic_vector(3 downto 0) := "0000";
  signal tmp: std_logic_vector(3 downto 0) := "0000";
begin
  -- 十位计数器需要四个触发器辅助
  dt0: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(0), q=>res(0),
nq=>tmp(0));
  dt1: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(1), q=>res(1),
nq=>tmp(1));
  dt2: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(2), q=>res(2),
nq=>tmp(2));
  dt3: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(3), q=>res(3),
nq=>tmp(3));

  -- 若时钟改变或重置
  process(clk, rst)
  begin
    -- 若为重置则
    if (rst = '1') then
      cnt <= "0001";
      car <= '0';
    -- 否则若时钟到达上升沿
    elsif (clk'event and clk = '1') then
      -- 若当前计数为 8，则提前传出进位
```

```
      if (cnt = "1000") then
        car <= '1';
        cnt <= cnt + 1;
      -- 若当前计数为 9，则下一次输出为 0，进位重置为 0
      elsif (cnt = "1001") then
        cnt <= "0000";
        car <= '0';
      -- 否则正常操作即可
      else
        cnt <= cnt + 1;
        car <= '0';
      end if;
    end if;
  end process;

end bhv;
```

工作原理：每次时钟到达上沿时（这里是点击一次 **clk**），则计数器加 1，当计数器计到 8 时，则进位置为 1（需要等待时钟周期，故需要提前传出）；当计数器计到 9 时，则计数器置零，进位也置为 0。

2.4 六位计数器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter6 is
port(
  clk, rst, car: in std_logic;
  res: buffer std_logic_vector(3 downto 0)
);
end counter6;

architecture bhv of counter6 is
  component dtrigger is
    port(
      cp, rst, d: in std_logic;
      q, nq: buffer std_logic
    );
  end component;
  signal cnt: std_logic_vector(3 downto 0) := "0000";
  signal tmp: std_logic_vector(3 downto 0) := "0000";
begin
  -- 六位计数器需要三个触发器辅助，为了元件的输出格式统一，这里还是使用 4 个触发器
```

```
  dt0: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(0), q=>res(0), nq=>tmp(0));
  dt1: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(1), q=>res(1), nq=>tmp(1));
  dt2: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(2), q=>res(2), nq=>tmp(2));
  dt3: dtrigger port map(cp=>clk, rst=>rst, d=>cnt(3), q=>res(3), nq=>tmp(3));

   -- 若时钟改变或重置
  process(clk, rst)
  begin
    -- 若为重置则
    if (rst = '1') then
      cnt <= "0000";
    -- 否则若时钟到达上升沿
    elsif (clk'event and clk = '1' and car = '1') then
      -- 若当前计数为 5，则下一次输出为 0，进位重置为 0
      if (cnt = "0101") then
        cnt <= "0000";
      -- 否则正常操作即可
      else
        cnt <= cnt + 1;
      end if;
    end if;
  end process;

end bhv;

```

工作原理：每次时钟到达上沿时（这里是点击一次 **clk**），则计数器加 1，当计数器计到 5 时，则计数器置零。

2.5 60 进制计数器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter is
port(
  high: out std_logic_vector(6 downto 0);
  low: out std_logic_vector(6 downto 0);
  clk, rst: in std_logic
);
```

```
end counter;

architecture bhv of counter is
  component counter6 is
  port(
    clk, rst, car: in std_logic;
    res: buffer std_logic_vector(3 downto 0)
  );
  end component;
  component counter10 is
  port(
    clk, rst: in std_logic;
    res: buffer std_logic_vector(3 downto 0);
    car: buffer std_logic
  );
  end component;
  component digit7 is
    port(
      key: in std_logic_vector(3 downto 0);
      output: out std_logic_vector(6 downto 0)
    );
  end component;
  signal keyh: std_logic_vector(3 downto 0) := "0000";
  signal keyl: std_logic_vector(3 downto 0) := "0000";
  signal car:std_logic;
begin
  -- 六位计数器和十位计数器
  c10: counter10 port map(clk, rst, keyl, car);
  c6: counter6 port map(clk, rst, car, keyh);
  -- 七位数码管输出
  dph: digit7 port map(keyh, high);
  dpl: digit7 port map(keyl, low);
end bhv;

```

工作原理：将 **clk** 和 **rst** 传入计数器之中，同时检测十位计数器返回的进位，并传入六位计数器中。除此之外，也需要将计数器的数值输出到七位数码管中。

2.6 秒表

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

```

```vhdl
entity clock is
port(
  high: out std_logic_vector(6 downto 0);
  low: out std_logic_vector(6 downto 0);
  clk, rst, pau: in std_logic
);
end clock;

architecture bhv of clock is
  component counter is
    port(
      high: out std_logic_vector(6 downto 0);
      low: out std_logic_vector(6 downto 0);
      clk, rst: in std_logic
    );
  end component;
  -- stop 为 1 停止计数，为 0 则继续计数
  signal stop: std_logic := '0';
  signal tmp: std_logic := '0';
  signal cnt: integer := 0;
begin
  ctr: counter port map(high, low, tmp, rst);
  -- 若点击停止键
  process(pau)
  begin
    if (pau'event and pau = '1') then
      -- 若 stop 为 1 则置为 0
      if (stop = '1') then
        stop <= '0';
      else
      -- 若 stop 为 0 则置为 1
        stop <= '1';
      end if;
    end if;
  end process;

  process(clk, rst, pau)
  begin
    -- 若点击重置键
    if (rst = '1') then
      cnt <= 0;
      tmp <= '0';
    -- 否则若 clk 被点击且 stop 为 0
    elsif (clk'event and clk = '1' and stop = '0') then
```

```
50        -- 若计数达 1000000，则时钟置为 1
51        if (cnt = 1000000) then
52          cnt <= 0;
53          tmp <= '1';
54        -- 若计数达 500000，则时钟置为 0
55        elsif (cnt = 500000) then
56          cnt <= cnt + 1;
57          tmp <= '0';
58        -- 其余情况不变
59        else
60          cnt <= cnt + 1;
61        end if;
62      end if;
63    end process;
64  end bhv;
65
```

工作原理：分别检查停止键、重置键和 **clk** 键，若为停止键则改变停止状态。若为重置键则重置状态，若为 clk 键则检查停止状态，若为计数状态则通过计数器的数值改变时钟值。

3 电路功能测试

3.1 实际操作

3.1.1 实验用具

本次使验使用了数字逻辑实验平台中的**无译码数码管**，一个**可编程模块**和一个**触发器**。

3.1.2 实验步骤

根据书上的端口设计接线。然后分别测试：

1. 等待时钟由 00 跳至 59，再回归 00；
2. 点击停止键，计数停止；
3. 点击重置键，计数清零，且数值保持为 00；
4. 再次点击停止键，计数开始；
5. 再次点击重置键，计数清零，然后开始计数；

至此，代码和接线均无误。

3.2 仿真实验

3.2.1 实验步骤

实验开始前将代码中的 1M 改为 2， 500K 改为 1，然后将 clk 设置为 1ns 的方波，再把 pau 和 rst 在适当的时机改为 1，便可得到结果。

3.2.2　　仿真结果

随意选择可以得到如下结果：

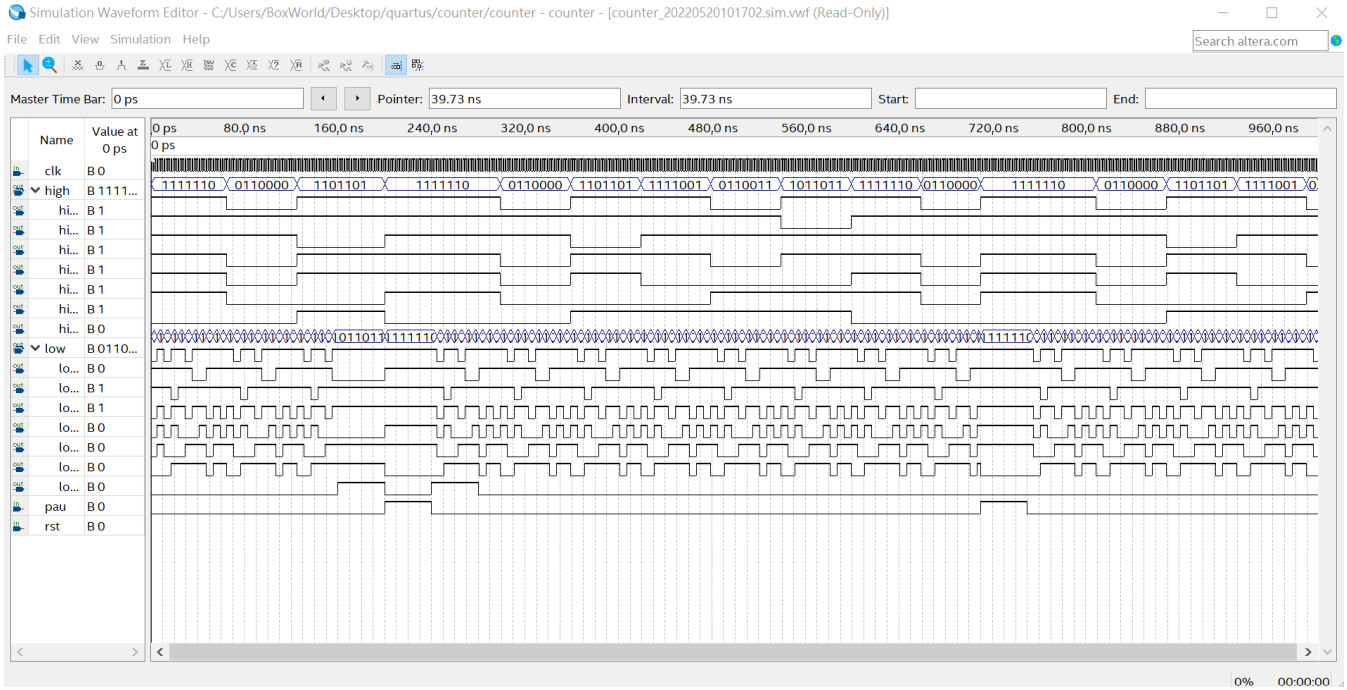

图1　秒表仿真结果

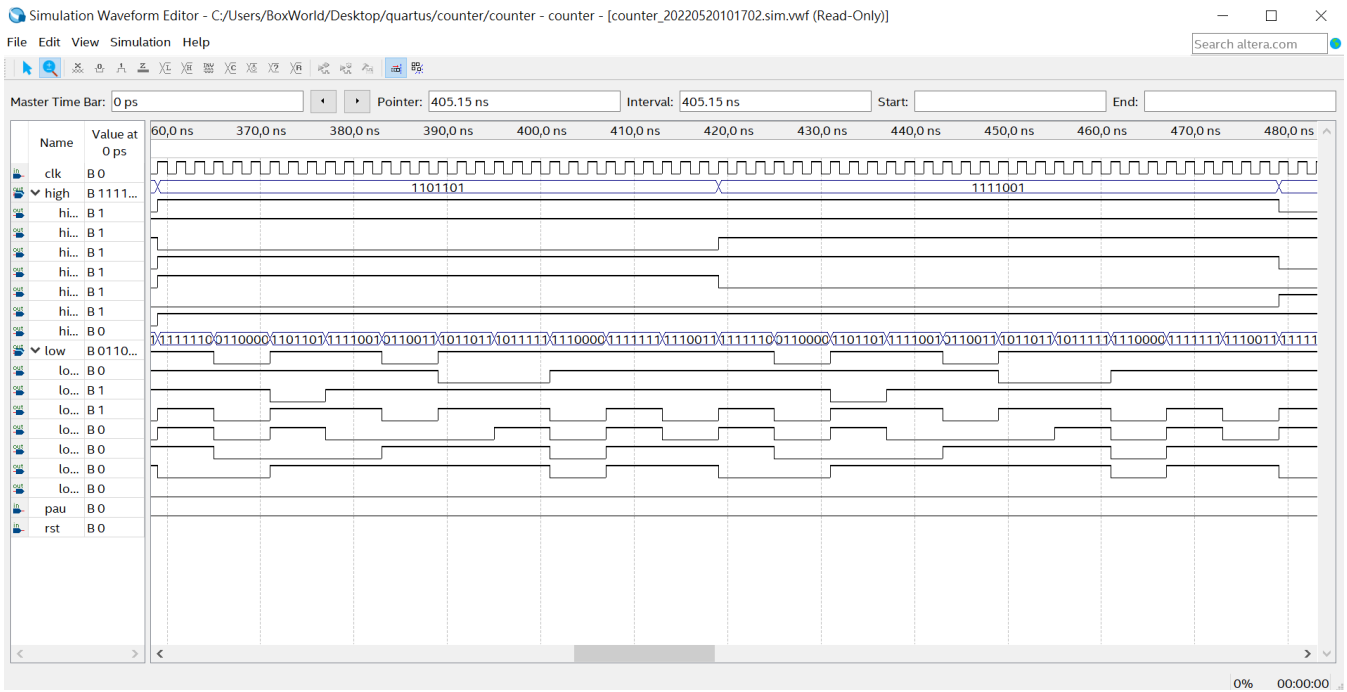

图2　秒表仿真结果（20–39）

可以看见，秒表在停止时可以正确停止，重置键也能发挥其作用。

4 遇到的问题与解决方法

在实验过程中，计数器的高位不能正常输出数字。经过一番观察后，发现原因是工作过程中直接使用进位做 CP 会导致过种不可预知的问题，因此需要保留 clk 作时钟，而进位作辅助便可解决问题。