

计数器的设计 实验报告

赵晨阳 2020012363

实验目的

1. 掌握时序逻辑电路的基本设计和分析方法
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和分析

实验内容

1. 实现带复位的 60 进制计数器
2. 使用 1MHz 时钟, 将计数器改成秒表
3. 在秒表中使用开关控制秒表启动和暂停

实验原理与代码分析

正边沿 D 触发器

```
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 myDFF is
7  -- 正边沿D触发器
8      port(
9          D, CP, Rd, Sd: in std_logic ;
10         Q: out std_logic := '0';
11         nQ: out std_logic := '1'
12     );
13 end myDFF;
14
15 architecture arch of myDFF is
16 begin
17     process(Rd, Sd, CP)
18     begin
19         if (Rd = '0') then
20             Q <= '0';
21             nQ <= '1';
22         elsif (Sd = '0') then
23             Q <= '1';
24             nQ <= '0';
25         elsif (CP'event and CP = '1') then
26             Q <= D;
27             nQ <= not D;
28         end if ;
29     end process ;
```

```
30 end arch;
```

D 为输入信号，CP 为时钟信号，Rd 为清零信号，Sd 为置位信号，Q 为正输出，Q_n 为负输出。当清零信号为‘0’时，将 Q 置零，Q_n 置一。当置位信号为‘0’时，将 Q 置一，Q_n 置零。否则当 CP 出现上升沿时，Q 赋值为 D，Q_n 赋值为 Q 的非。

六进制加法计数器

```
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 clock_6 is
7      port(
8          clk, Rd, c0: in std_logic ;
9          out_6: out std_logic_vector(3 downto 0)
10     );
11 end clock_6;
12
13 architecture clock_6_bhv of clock_6 is
14     component myDFF is
15         port(
16             D, CP, Rd, Sd: in std_logic ;
17             Q: out std_logic := '0';
18             nQ: out std_logic := '1'
19         );
20     end component;
21     signal D, Q, nQ: std_logic_vector(2 downto 0);
22 begin
23     lb0: for i in 0 to 2 generate
24         Di: myDFF port map(D(i), clk , Rd, '1', Q(i), nQ(i));
25     end generate lb0;
26     D(0) <= c0 xor Q(0);
27     D(1) <= ((not c0) and Q(1)) or (c0 and (nQ(2) and (Q(1) xor
Q(0))));
```

```

28     D(2) <= ((not c0) and Q(2)) or (c0 and ((Q(2) and nQ(0)) or
(Q(1) and Q(0))));
29     out_6(2 downto 0) <= Q;
30     out_6(3) <= '0';
31 end clock_6_bhv;

```

使用同步计数器, clk 为时钟信号, Rd 为清零信号, c0 为低位进位信号, out_6 为输出计数结果, 范围在 0000 与 0101 的闭区间内。

十进制加法计数器

```

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 clock_10 is
7      port(
8          clk , Rd: in std_logic ;
9          out_10: out std_logic_vector(3 downto 0)
10         );
11  end clock_10;
12
13  architecture clock_10_bhv of clock_10 is
14      component myDFF is
15          port(
16              D, CP, Rd, Sd: in std_logic ;
17              Q: out std_logic := '0';
18              nQ: out std_logic := '1'
19          );
20      end component;
21      signal D, Q, nQ: std_logic_vector(3 downto 0);
22  begin
23      lb0: for i in 0 to 3 generate
24          dt: myDFF port map(D(i), clk , Rd, '1', Q(i), nQ(i));
25      end generate lb0;

```

```

26     D(0) <= nQ(0);
27     D(1) <= (nQ(0) xor nQ(1)) and (Q(0) nand Q(3));
28     D(2) <= ((not Q(0)) or (not Q(1))) xnor Q(2);
29     D(3) <= ((not ((nQ(0) or nQ(1) or nQ(2))) xnor nQ(3)) and
(Q(0) nand Q(3)));
30     out_10 <= Q;
31 end clock_10_bhv;

```

使用同步计数器，clk 为时钟信号，Rd 为清零信号，out_10 为输出计数结果，范围在 0000 与 1001 的闭区间内。

六十进制计数器

```

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 clock_60 is
7      port(
8          clk , rst : in std_logic ;
9          low, high: out std_logic_vector(3 downto 0)
10     );
11 end clock_60;
12
13 architecture clock_60_bhv of clock_60 is
14     component myDFF
15         port(
16             D, CP, Rd, Sd: in std_logic ;
17             Q: out std_logic := '0';
18             nQ: out std_logic := '1'
19         );
20     end component;
21
22     component clock_6
23         port(

```

```

24     clk , Rd, c0: in std_logic ;
25     out_6: out std_logic_vector(3 downto 0)
26 );
27 end component;
28
29 component clock_10
30 port(
31     clk , Rd: in std_logic ;
32     out_10: out std_logic_vector(3 downto 0)
33 );
34 end component;
35
36 signal D, c0, Rd: std_logic ;
37 signal QL, QH: std_logic_vector(3 downto 0);
38 begin
39     count_low: clock_10 port map(clk, Rd, QL);
40     count_high: clock_6 port map(clk, Rd, c0, QH);
41     trriger: myDFF port map(D, clk, Rd, '1', c0);
42     Rd <= not rst;
43     D <= (QL(3) and (not QL(2))) and ((not QL(1)) and (not
44 QL(0)));
45     low <= QL;
46     high <= QH;
47 end clock_60_bhv;

```

输入 clk 为时钟信号，rst 为复位信号，输出 low 为低位二进制结果，high 为高位二进制结果。

译码器

```

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 digit_7 is

```

```

7   port(
8       number: in std_logic_vector(3 downto 0);
9       display: out std_logic_vector(6 downto 0)
10  );
11 end digit_7;
12 architecture arch of digit_7 is
13 begin
14     process(number)
15     begin
16         case number is
17             when "0000"=>display<="1111110";
18             when "0001"=>display<="0110000";
19             when "0010"=>display<="1101101";
20             when "0011"=>display<="1111001";
21             when "0100"=>display<="0110011";
22             when "0101"=>display<="1011011";
23             when "0110"=>display<="0011111";
24             when "0111"=>display<="1110000";
25             when "1000"=>display<="1111111";
26             when "1001"=>display<="1110011";
27             when "1010"=>display<="1110111";
28             when "1011"=>display<="0011111";
29             when "1100"=>display<="1001110";
30             when "1101"=>display<="0111101";
31             when "1110"=>display<="1001111";
32             when "1111"=>display<="1000111";
33             when others=>display<="0000000";
34         end case;
35     end process;
36 end arch;

```

参考点亮数字人生，输入二进制数 nin，输出可经不带译码器的数码管进行输出的译码后的结果。

带译码的六十进制加法计数器

```
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 clock is
7      port(
8          clk , rst : in std_logic ;
9          lout , hout: out std_logic_vector
10         );
11  end clock;
12
13  architecture clock_bhv of clock is
14      component clock_60
15          port(
16              clk , rst : in std_logic ;
17              low, high: out std_logic_vector(3 downto 0)
18          );
19  end component;
20  component digit_7
21      port(
22          number: in std_logic_vector(3 downto 0);
23          display: out std_logic_vector(6 downto 0)
24      );
25  end component;
26  signal low, high: std_logic_vector(3 downto 0);
27  begin
28      count: clock_60 port map(clk, rst, low, high);
29      digit_low: digit_7 port map(low, lout);
30      digit_high: digit_7 port map(high, hout);
31  end clock_bhv;
```

使用普通 60 进制加法计数器进行计数，将输出经过译码器转化为译码后的输出。

```
1  library ieee;
```

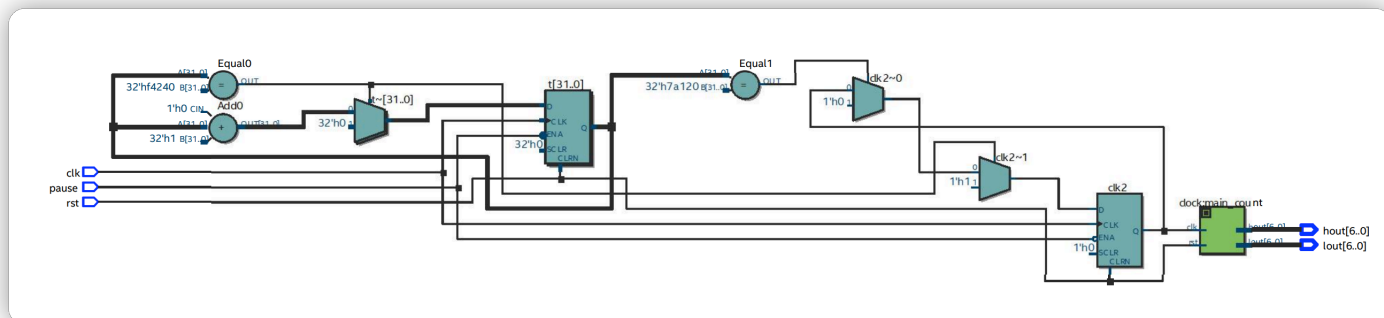
```

2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity count is
7      port(
8          clk , rst , pause: in std_logic ;
9          lout , hout: out std_logic_vector(6 downto 0)
10     );
11 end count;
12
13 architecture count_bhv of count is
14     component clock
15     port(
16         clk , rst : in std_logic ;
17         lout , hout: out std_logic_vector(6 downto 0)
18     );
19 end component;
20 signal clk2 : std_logic := '0';
21 signal t: integer := 0;
22 begin
23     main_count: clock port map(clk2, rst , lout , hout);
24     process(pause, clk , rst )
25     begin
26         if ( rst = '1') then
27             t <= 0;
28             clk2 <= '0';
29         elsif (clk 'event and clk = '1' and pause = '0') then
30             if (t = 1000000) then
31                 t <= 0;
32                 clk2 <= '1';
33             elsif (t = 500000) then
34                 t <= t + 1;
35                 clk2 <= '0';
36             else
37                 t <= t + 1;
38             end if ;
39         end if ;
40     end process ;
41 end count_bhv;

```

输入 clk 为时钟信号, rst 为复位信号, pause 为暂停信号, 输出 lout 为低位译码输出, hout 为高位译码输出。clk 接到实验块上的 1MHz 脉冲信号上, 便可实现自动的 60 进制正计时秒表。

电路结果



仿真结果

仿真时，将时钟信号的时间跨度缩减为 10(原本的时间跨度为 1000000，仿真软件的精度不足以实现如此高频率的脉冲信号)，模拟暂停，复位等功能，仿真结果正确。

